For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex libris
universitatis
albertaensis

BRUCE PEEL SPECIAL COLLECTIONS LIBRARY
UNIVERSITY OF ALBERTA LIBRARY

REQUEST FOR DUPLICATION

Digitized by the Internet Archive
in 2023 with funding from
University of Alberta Library

https://archive.org/details/Taugher1983

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR [illegible]

TITLE OF THESIS AN EXPLORATORY INVESTIGATION [illegible]

[illegible]

DEGREE FOR WHICH THESIS WAS PRESENTED [illegible]

YEAR THIS DEGREE GRANTED FALL [illegible]

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

[illegible]

PERMANENT ADDRESS:

[illegible]

[illegible]

[illegible]

DATED [illegible]

THE UNIVERSITY OF ALBERTA

An Efficient Representation for Time Information

by

© James Edwin Taugher

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF Master of Science

Department of Computer Science

EDMONTON, ALBERTA
Fall 1983

Abstract

Special purpose mechanisms are required to augment general knowledge representations if they are to allow efficient deduction over a wide range of "everyday" topics. Such a special purpose mechanism for representing and reasoning about time is proposed. An acyclic directed graph is constructed to represent time points and events, time order information relating these, and dates and durations. A method of question answering is presented whose time complexity is nearly independent of the number of time facts stored and requires storage proportional to the number of time facts plus relations between these.

ABSTRACT

Special purpose mechanisms are required to support general knowledge representations if they are to allow efficient inference over a wide range of "everyday" topics. Such a special purpose mechanism for representing and reasoning about time is proposed. An acyclic directed graph is constructed to represent time points and events, time order information relating these, and dates and durations. A method of question answering is presented whose time complexity is nearly independent of the number of time facts stored and requires storage proportional to the number of time facts plus relations between these.

Acknowledgements

First and foremost I would like to thank my supervisor Len Schubert, for his guidance and assistance throughout the course of this research. The criticism and many of the key ideas that he provided, as well as the patience he displayed were prime factors in my successful completion of this work.

I would also like to thank the members of my committee Bernard Linsky, Jeff Pelletier and William Armstrong. Their comments and insights were both helpful and appreciated.

Finally I would like to thank my family and friends for the various forms of assistance that they have given me.

Table of Contents

List of Tables

List of Figures

1. Introduction

This thesis is concerned with the design and implementation of an efficient method for reasoning about temporal knowledge. This time handler has been developed as a special purpose inference mechanism, to aid more general, higher-order inference mechanisms.

Special purpose inference methods are necessary to support a general knowledge representation and reasoning system that has been under development at the University of Alberta for a number of years, (Covington and Schubert [1979], Schubert and Goebel [1980], Schubert and Papalaskaris [1981]). This system accepts information in the form of assertions in modal predicate logic and organizes it within the framework of a semantic net. The system can perform concept and topic oriented retrieval, as well as property and relationship inheritance. A resolution-based deductive question-answering algorithm operates on the knowledge base (DeHaan and Schubert, in preparation).

The general question answering algorithm can answer many simple questions quickly, irrespective of the knowledge-base size. This is made possible by highly selective retrieval mechanisms; for example, given that Clyde is an elephant and elephants are known to be grey, the system easily answers "Is Clyde grey?". However, there are certain kinds of questions which still cannot be done quickly. Among these are questions which implicitly involve "chains" of inferences over types, parts, and times.

In order to answer questions without noticeable delay, or at least no greater delay than would be expected with a human conversational partner, it is necessary to augment the general resolution theorem prover with special purpose mechanisms. These mechanisms provide fast inferences within a particular class of relations and can greatly enhance the speed of the general resolution process. A number of such inference mechanisms have been developed to date.

Special purpose mechanisms are concerned with extending inference capabilities in two possible ways; evaluation of propositions and extending the types of allowable resolutions. The special purpose mechanism described here will provide proposition evaluation for literals involving time, for example *After(a, b)*. Other special purpose inference mechanisms already developed within the University of Alberta project include mechanisms for parts, types, and colours (Papalaskaris and Schubert [1981], Schubert et al [1983]).

A brief description of special purpose inference mechanisms and a closer look at the types of information and inferences that both can and cannot be represented in this time handler are given in the following sections of this chapter.

Chapter 2 critically examines other time handling mechanisms described in the literature, outlines the constraints on a satisfactory representation and describes some preliminary research done by this author laying the

foundation for the final design. Chapter 3 provides an overview of both the structures and the retrieval and inference mechanisms. Details of the representation and the algorithms for construction are given in chapter 4, while chapter 5 provides the details of the retrieval and inference techniques, and provides theoretical and empirical complexity analysis results. Chapter 6 contains the conclusions and suggests avenues of future research.

1.1 Special Purpose Inference Mechanisms

The special inference methods of interest here are those which are *fast* and *domain-independent*. A method is fast if it is at least several times faster than the unaided general reasoner making the same inferences. It is domain-independent to the extent that it can aid the general reasoner in a broad range of particular domains. (Hence, mechanisms designed exclusively for special applications such as computer chess, circuit design, or chemical analysis are not under consideration.)

There are several special classes of relations that cut across many domains, yet lend themselves to efficient handling by special methods. These include relations among parts (such as part-of and disjoint-from), relations among types (such as subtype-of and incompatible-with), and relations among times. For example, inferring the part-of relation between "thumb and arm", or "New York City and North America", should be trivial; yet, without special

methods they may be computationally expensive. Similarly, determining the temporal order of two events such as "the Apollo 11 mission", and "the construction of the first computer", could consume large computational resources by a resolution prover (or other uniform inference method), although people seem to have no problem with such relations.

These special classes of relations were chosen for special methods because of their importance in many domains and because of the ease with which humans handle them (Schubert et al [1983]). The emphasis is on the design of comprehensive and efficient mechanisms for dealing with each class of relations. Schubert [1979] and Papalaskaris and Schubert [1981], provided the foundations for this research by developing a general tool (the P-graph) for special purpose inference, which is discussed further in 2.3.

Since these special methods are targeted for inclusion within the framework of a larger system, there are different areas of emphasis compared to a functionally similar stand-alone system. In particular these systems will be consulted for quick advice or information for particular kinds of relations. Therefore these mechanisms must be able to reply quickly, even if no answer has been found. Also, it is desirable that the processes controlling the larger system be able to specify levels of effort to be expended by the special methods in deriving an answer to a query. This allows for flexible control of the special methods by the general reasoning system, which has access to much more

information (of all types).

1.2 Time Information and Relations

This section describes the type of information that needs to be represented in this temporal reasoner, as well as the types that will not be considered here. This can be broken into three distinct types; relative order, dates, and durations. For completeness all three should be handled by the special purpose inference mechanism.

Relative order refers to the relative positioning of events with respect to time. An example of information of this type is, "John saw Sue *after* he left the office, but *before* he got to his car". In this example, the first event, "John seeing Sue" is ordered later in time relative to the second event mentioned, "John leaving his office". The third event, "John gets to his car" is ordered after both other events. Besides the time relations, "before" and "after", other important relations between events are "coincident", "during", or "overlapping".

The relations incorporated in this project are the ones commonly encountered in all kinds of stories and dialogue. But this is in no way a complete list of all possible time relations that could be defined over events or that could be represented by this time handler. There is a potentially very large set of such relations (such as "long-before", "just-before", "consecutive", etc.). All relations that can be expressed as a single, consistent set of pairs of event

end points ordered by the relation '$\leq$' can be represented within the time graph presented here. Other relations including logical operators such as negation (for example "not during E_1"), disjunction (for example "... either before E_1 or before E_2"), repetitive events (for example "Every week the car breaks down."), and probabilities (for example "probably after E_2") cannot be represented in this time handler. Such relations can still be represented within the scope of the entire system. The way such relations would be handled by the semantic net, and how this time mechanism interacts with the main net is described in the following section.

Dates, or absolute times as they will be referred to throughout the remainder of this thesis, consist of a specific segment of some[1] chronological scale. An absolute time may be fully specified such as, "The clock struck twelve noon on December 7, 1942.", or more commonly only partially specified as in, "I saw John last July". Whether fully or partially specified, the absolute time information must be recorded.

Durations are the final type of information. This refers to the length of time between different events, or between the beginning and end of a single event. An example of durational information is, "John broke his leg ten days ago". In this case the event is given a duration of ten days before the time of speech. As with absolute times, durations

---

[1] Normally this will be the Gregorian Calendar but this may differ for some fairy tales, fictional stories, etc.

are commonly specified to varying degrees, and all information contained in incompletely specified durations must be included in the representation.

Completeness requires that all types of information be kept in the time handler; however, human cognitive abilities are not equal in making inferences across all of these types. Determining the relative order of events is usually easier than determining a specific date for some event, or determining the elapsed time between events. For this reason the time representation which is developed in this thesis emphasizes fast[2] inference of the relative order of events.

1.3 Interaction with the Semantic Net

This section describes how the time specialist outlined in this thesis interacts with the semantic net. Since the purpose of the special purpose inference mechanism is to provide quick inference of a specialized sort, not all inferences involving time are made within the time mechanism. With respect to the overall system the time mechanism will be involved in both the information input and the question answering processes.

When facts are presented to the general system which involve instances of particular events related by a time predicate such as $\mathit{After}(E_1, E_2)$, this information is recognized (by its form) as appropriate for the time mechanism. Similarly, if a disjunction involving a literal

---

[2] Constant time if possible.

with a time predicate is presented such as Before(E_1, E_3) V Before(E_2, E_1), the special purpose time mechanism can be consulted to determine if any of the literals can be evaluated. In this example since Before(E_2, E_1) would be detected 'true' by the time handler, the entire clause can be discarded (without adding it to the main net) since it adds no information.

During question answering a number of clauses might be generated for resolution. Suppose one such clause is P(a, b) V During(E_1, E_7) V R(z). Again the literal During(E_1, E_7) is recognized as being the appropriate form for the special purpose time mechanism which is then consulted to attempt to evaluate this literal. If this literal is evaluated to true, then the entire clause can be eliminated. If the literal is evaluated to false, then the literal itself can be removed from the clause leaving P(a, b) V R(z). In either case the number of literals (and possibly clauses) has been reduced possibly eliminating many inference steps which would otherwise be required and thus allowing faster completion of the proof.

Sentences involving universally quantified variables are incorporated exclusively within the main net. For example consider the sentence, "John starts work only after he arrives at the office." Translated into predicate calculus notation, this would be

∀X,∀T,∀S [[[John arrives-at T Office1] &
[John starts-work S] & [X day] & [T during X] &

[S during X]] ⇒ [T before S]][3]

This rule is then stored in the semantic net. Whenever particular instantiations of the premises are supplied, the events they constitute would be stored in the time handler.

Such instantiations might supply a particular day A, arrival E_1 and start of work E_2. The two events representing John's arrival at the office on day 'A' (E_1), and his starting work on the same day (E_2) would be communicated to the time handler. If subsequently the main inference system supplied the above rule to the instanitated premises, obtaining the conclusion [E_1 before E_2], this inferred fact would in turn be presented to the time handler. Thus, time information in both the premises and the conclusion would then be available to the time handler, for use in making further inferences in conjunction with the rest of its collection of time relations.

In this way general (universally quantified) knowledge about events and actions can in principle be handled by the semantic net (see Covington and Schubert [1980], and deHaan (in preparation) for the current status of the inference system).

---

[3] Where X, T, S can be thought of as event variables or time-interval variables; the predications are in infix form, i.e. the first argument precedes the predicate and the rest (if any) follow it.

2. Representation of Temporal Knowledge

The problems and complexities involved in designing a general knowledge representation system are immense, and because of this a number of researchers have elected to tackle knowledge representations in limited domains as a starting point, with the intention of later expanding the scope of their systems. However, by taking advantage of simplifications available with a limited domain, many have sacrificed the extensibility of their systems. In this chapter a survey of other research in temporal knowledge is made. A number of these systems suffer from this sort of problem. Later sections of this chapter examine the constraints on a satisfactory representation and a number of designs are attempted. Finally, empirical examination of some stories is done to derive some characteristics on which to base our representation.

2.1 Handling Time Information and Inferences

In the past few years there has been increasing recognition of the importance of temporal information in any knowledge representation system. In view of this, there have been a number of systems designed to handle time. Most of these systems are stand-alone, but some operate within the framework of a larger, more general representation system.

The work can generally be separated into two groups. One group approaches the problem from a linguistic viewpoint. These systems are designed to analyze the

temporal references in English or English-like sentences. The second group for the most part takes for granted the initial interpretive phase, and focuses on the representation and manipulation of time information. Although the work presented in this thesis belongs to the second group, previous work from both areas will be reviewed, since they all contribute to the current body of knowledge about handling time.

Findler and Chen [1971] provided one of the earliest attempts to deal with the representational issues of time. Two important concepts are discussed in their paper. First the distinction between relevant, irrelevant, and partially specified chronological data can be made within their system. Secondly, they allow symbolic names to be associated with specific time points giving great flexibility to the association of information with events. These are features which, although appearing early in the research of the field, are often overlooked in later developments.

Findler and Chen make an unnecessary distinction between "point" and "duration" events, which forces them to externally categorize all events. Although the complexity of their system is not mentioned, they present a "moderately sized data base" consisting of six events and relations between these, and suggest that reasonable question answering times[4] are evidence that their methods are not overly complex. However, the hundreds of events in even a

---

[4] Up to forty seconds of CPU(?) time on a CDC-6400 for one question.

simple fairy tale might cause efficiency problems, in light of their example with six events.

Bertram Bruce [1972] deals mostly with the problem of extracting time references from English sentences. His analysis of tense and temporal references was made with the goal of developing a suitable semantic representation. In his subsequent work on the representation system Chronos (Bruce [1972] and Bruce and Singer [1973]), he found major shortcomings in his original system, because of a lack an adequate method for representing incomplete or partially specified times. He does make clear through precise definitions the concepts of time as an ordered set of time points, time-segments, and various time-segment relations.

The Chronos system was the implementation of Bruce's work, and it extracts time information from English sentences by looking for certain "trigger words" (such as "yesterday", "afternoon", "morning", etc.). This time information is stored in a list representing year, month, day, hour and minute. There is no symbol for unspecified portions of the date, so he uses rather dubious defaults to fill these slots. (For example, given just the year of some event, his representation fills the other slots with the first month of the year, the first day of the month, zero hours and zero minutes. So from a partially defined date he produces an arbitrary specific date.). The retrieval mechanisms use exhaustive search techniques on "event trees" to find particular events and their relations, requiring

question answering times proportional to the number of facts stored.

Undoubtedly the most complete work on temporal knowledge was the "time specialist" of K. Kahn. His work, described in Kahn [1975] and Kahn and Gorry [1977], was an attempt to deal with all types of temporal information. In order to accommodate the different types he creates several different structures each responsible for a different type of information. While he strives for completeness, by taking each type of time information and creating distinct structures and methods for accessing these, his system suffers from the lack of a single representation, which Kahn suggests can not be powerful enough.

Kahn's time-specialist is based on concepts and ground work laid out by Bruce, and Findler and Chen. Communication with the system is through a special input syntax for expressing properties and relationships of events. These are used both to enter facts into the system (for example, (TIME-OF (EVENT-A) (AFTER (EVENT-B)))), and to ask questions (for example, (TIME-OF ? (AFTER (EVENT-B)))). A serious fault with the system was that it was left to the user to decide the internal representation that the time specialist would utilize based on the type of temporal information and questions that were expected. This is in some sense leading the system by the hand, since the user must pre-analyze the input, determine the prominent temporal characteristics, and then decide on the appropriate time-specialist

representation to handle these characteristics. Kahn may have forseen this objection with his system, because he provides an interface capable of transforming a story represented in one structure into an equivalent representation in another structure. But a single representation is sufficient and avoids all of the transformation problems, as will be seen.

He addresses the problem of incompletely specified dates. An upper and lower bound on a date is inferred from the partially specified one, and this interval is associated with a particular event. Unlike Findler and Chen however, he does not provide any capability for using symbolic names to represent explicit times.

An elaborate system for incorporating approximate dates and durations, referred to as "fuzz" is provided. For example, the duration "about three weeks", can be equivalently represented in his system as:

```
1). (BY-AMOUNT (WEEKS 3) (FUZZ (DAYS 7)))
2). (INTERVAL (WEEKS 2) (WEEKS 4))
3). (FUZZY-AMOUNT (ABOUT A-FEW-WEEKS))
4). (FUZZY-AMOUNT (A-BIT-MORE-THAN A-HALF MONTHS))
```

This does not exhaust all of the possibilities, the actual form being dependent on the input form. Kahn's system provides various methods for determining equivalence between these widely disparate forms.

A shortcoming of Kahn's question answerer is the reliance on two exhaustive search techniques if other

methods of question answering fail. The first method is a breadth-first search of events from a given starting event. This could incur time proportional to the number of events plus the number of relations between events that are in the system. The second method searches through all events in the system and can require time complexity comparable to the breadth-first search. If either (or worse, both) of these methods are employed in an application with many events and relations, they could consume large computational resources for answering a single, perhaps unimportant question.

Kahn's system was used as the representational basis for a thesis by Robin Cohen [1977] in which she extracted the temporal references from English sentences and then used these as input to a slightly modified version of Kahn's time specialist. Her thesis focuses on the linguistic analysis of temporal references, based on a linguistic theory originally proposed by H. Reichenbach.

Cohen identifies Kahn's multiple structures as the major problem with his work, but her solution has major problems as well. She modifies Kahn's system to force everything onto an ordered time line. Since most events that are encountered in sentences do not have explicit dates associated with them, she is forced into proposing complicated "guessing" heuristics to place events on her date-line representation. As well, since her primary concern was with linguistic analysis of temporal references, efficiency considerations for her algorithms are largely

ignored.

More recent efforts on temporal knowledge include the work of James Allen [Jan., 1981], [Nov., 1981] and [1983], concerned with representation of temporal intervals and the use of a temporal logic in planning. Allen stresses the need for representing temporal intervals only, as opposed to both intervals and instantaneous point events. He starts by defining a set of nine relations which can hold between pairs of time intervals. His system creates a temporal interval network, with pairs of intervals connected by edges with zero or more of nine possible relations indicated on them. If there is uncertainty about a relationship, then all possible relations which may apply are entered on the edge. When some new information is added to the graph, all of the consequences must be determined. To achieve this, Allen adds the new fact to the network and then calculates any new relationships that may be inferred from this information. These new relationships are then in turn entered in the network, with each of them having possible consequences and so on.

Although the complexity of his algorithms are not explicitly stated, an estimate can be obtained from his descriptions. The network might be fully connected and every relation might require updating as a result of adding a single new fact. In such a worst case scenario, the algorithms for adding the new information have time requirements that increase with the square of the number of

distinct time intervals stored. This complexity ignores the possibility of more than one time relationship between a given pair of time intervals, which would increase the time requirements.

A further problem with Allen's representation, which he tries to remedy, is the fact that every interval is explicitily related to every other interval. He attempts to exploit some properties of the during relation to reduce the number of connections, but is not entirely successful. As Allen admits, the resulting arbitrary graph structure might have to be searched to retrieve answers to queries, which would require time proportional to the number of time intervals plus the number of distinct relations between intervals. In general, Allen's work is more concerned with planning and using temporal logic to effect plans and actions, than it is with designing an efficient structure for storing and inferring temporal relations from temporal information.

Along similar lines to Allen's work is McDermott [1982] who also outlines a temporal logic for use in planning. As with Allen, McDermott is primarily concerned with augmenting a planning system with temporal information and reasoning. McDermott applies all processes and facts entering his reasoning system to his temporal logic, and draws inferences about persistence and causality which can then be used in planning.

Hirschman [1981] and Hirschman and Story [1981], describe a representation for time relations in narratives. The fact that they restrict the input text allows them to exploit some conventions of narrative discourse, such as default time relations between consecutive clauses and/or sentences. Hirschman introduces the concept of a time graph, with the edges in the graph representing the relation '≤', and the nodes representing either an event or the beginning or ending time point of an event. As with previous systems, she does not directly address the problem of efficiency, and every time a new event is mentioned the entire graph is searched in an attempt to resolve anaphora.

The time requirements to search the entire graph increase with the number of events and time points plus the number of relations between these. What makes the time requirements worse in her system, is that this computational effort is not a worst case figure, but must be expended for every addition to the graph, making it the best case complexity as well.

One recent contribution from the linguistic camp is by Mark Grover [1982]. Grover attempts to integrate results from investigations in linguistics, epistemological models of time, and computer-oriented models of time-based information. The input to Grover's system is English sentences which are then parsed using Montague grammar, modified by Dowty's translation rules. A time graph is utilized in which instances of generic events are stored on

the nodes. In the time graph used by Grover, different paths through the graph represent different possible worlds about which the system has some knowledge. Questions can be asked about the occurrence of events using a format similar to the input format. Grover employs a four-valued belief logic which permits four valid responses to input queries. Aside from true, false, and unknown, a fourth response implies that the system has recorded both the occurrence and non-occurrence of the event in different possible worlds.

In Grover's paper there is no explanation of the benefit of having an alternative possible worlds model for temporal information. While his attempt to incorporate ideas from different areas of artificial intelligence is a potentially fruitful approach, the only implemented portion of his work is the event representation, and this does not cover the areas of concern in this thesis.

Another recent linguistically oriented work is by Almeida and Shapiro [1983]. They are concerned with determining the temporal structure of narrative text, and examine in detail the roles of aspectual class and the progressive/nonprogressive distinction in determining temporal relations. The work presented in their paper is directed at parsing narrative text to achieve correct temporal relations for events based on these linguistic features. The representation of the information is based on Allen's work, discussed previously.

Another recent system developed for representing time information and making inferences is by Marc Vilain [1982]. Vilain's system records the time relation between two time intervals in a relational vector, where the relations are basically the same as those described by Bruce. A logic system involving composition rules for the primitive relational vectors is used to create new vectors which are deductions from the previous ones. Vilain's system appears to be impractical because the complexity is $O(n^3)$ in time and space, where n is the number of intervals about which assertions have been made.

E. Kandrashina [1983] describes a system called a "T-model" for representing temporal information. Included in this representation are definitions of point, interval, quantity and chain. Kandrashina's chains are sequences of time intervals which relate to one aspect of representing time information that has been largely neglected elsewhere, namely the representation of habitual or repetitive events. Special relations (such as "synchro-overlaps" and "alternation") are defined for representing various (regular and sporadic) interrelations between sequences of intervals. These relations will be useful for representing activities and processes, but only a small portion of the representation has been implemented. Since the paper is concerned almost exclusively with the representation (as opposed to use) of time information, it has little to say about the processing issues which are the main concern of

this thesis.

Malik and Binford [1983] describe a representation for both temporal and spatial information. The system converts relations between the endpoints of events into inequalities, and then uses linear programming to make inferences from these constraints. By basing their question answering on linear programming their system is formally adequate for questions which are themselves expressible as sets of linear inequalities, which provides an advantage over previous representations. Since they represent linear inequalities, all time information that can be expressed as such can be represented. These include time ordering ($t_1 \leq t_2$), bounds on absolute times ($1981 \leq t_2$ & $t_2 \leq 1983$), durations and bounds on durations ($t_2 - t_1 \leq 5$ & $t_2 - t_1 \geq 1$) and relative durations ($t_2 - t_1 \leq t_4 - t_3 + 7$, $t_4 - t_2 \leq 3(t_5 - t_1)$). Similarly, linear programming allows them to make all valid inferences (for example, $t_2 - t_1 = 2(t_4 - t_3)$) and check for inconsistency whenever a new inequality is added to the system.

In order to avoid the inefficiency of linear programming (obtaining a feasible solution by the Simplex method has worst case time requirements in **NP**.), they group related events into "clusters" and perform the linear programming techniques on these clusters. However, this requires them to provide "transformation" algorithms (cf. Kahn) to relate events in different clusters, and the complexity of these algorithms is never mentioned. Aside

from these concerns their paper represents one of the best attempts at time representation because of their emphasis on formal adequacy.

None of the systems that have been developed to date are concerned with the efficiency of their methods. A number of temporal systems discussed here are primarily focused on the linguistic processing of temporal expressions and hence are understandably less concerned with temporal inference. But if general language understanding and reasoning is to be achieved it will require massive amounts of stored information. In such circumstances, efficiency of both time and space are essential. The various problems that need to be solved in a time representation are covered among the various papers in the literature. However, combining the solutions into a complete, efficient solution to temporal knowledge representation, is still to be achieved.

2.2 Requirements for a Satisfactory Representation

A method is required for storing all the time information that is found in processing narrative texts of all kinds. Fast methods for retrieving the information and making inferences based on it are desired so that the time module will enhance the general reasoning system, and not slow it down. Processing of natural language discourse requires a large base of knowledge, so ideally inference algorithms independent of the size of the information base are sought. The methods presented in this thesis come close

to meeting these requirements.

Certain restrictions are placed upon this time handler since it is to be incorporated within a larger project. Aside from efficiency concerns, each new piece of information must be fully processed when it is entered so that queries, and additions of new information can be fully inter-leaved.

Further requirements of the representation are a need to represent all types of temporal information in a single unified structure. This is to avoid complications of choosing between various representations, such as those that arose with Kahn's system. The types of temporal information include relative (before/after) positioning, absolute times (dates) and durational information. There must be allowance for incomplete and partially specified dates, as well as a means of representing approximate time specifications.

Finally, it should be possible to specify the intensiveness of the effort to be made in satisfying a particular query. Since the time handler will be a part of a larger system, there will be occasions when a quick "unknown" response to a time query is preferable to a more informative, but costly answer. Such a mechanism will provide a means of external control over the time inference algorithms without any knowledge of the internal processes being employed.

A representation based on relationships among time points, rather than intervals, is favoured because it allows

a simpler design. An interval-based representation requires distinct representation for each interval of the various relations (such as before, consecutive, overlapping, during, etc.). However, a representation based on time points allows all the various event relations to be collapsed into one or two ordering ("before") relations among time points. This greatly eases the design of a simple time representation permitting efficient inference of relations among both time points and time intervals, represented in terms of their end points.

2.3 Application of P-Graphs

There has been much effort in recent years at the University of Alberta to develop efficient and complete methods for representing relationships among parts of objects (Schubert [1979], [1980], Papalaskaris [1982] and Papalaskaris and Schubert [1982]). Since time relationships also require special purpose inference methods, an attempt was made to adapt these techniques to the problem of time.

Schubert's examination of partitioning assertions led to the creation of a class of P-graphs for representing sets of partitioning assertions. A partitioning assertion has the form [T P t_1 t_2 ... t_i] and is interpreted as object T partitioned into disjoint parts t_1, t_2, ..., t_i. Closed and semi-closed P-graphs have been developed (Papalaskaris and Schubert [1983]) to permit efficient complete algorithms for answering part-of and disjointness questions.

By applying partitioning assertions to the time intervals occupied by events and adding a time ordering relation, the usual relations between events can be expressed in P-graph form. This would permit the time relation representation to utilize previously developed P-graph algorithms.

Although this approach seemed promising at first, there was a problem with the efficiency of these methods. It is the generality of the P-graph that allows it to be adapted to the representation of time relations, but this same generality prevents the P-graph from exploiting special features and properties of time information which result from the ordering relation. Since one of the prime concerns of this research was finding a representation that permitted efficient question answering, the P-graph representation was abandoned in favour of a directed graph of time points.

The first step of the new approach is to decompose all events into time points, and treat these as a partially ordered set with respect to time. A partial order '$\leq$' is defined on a given set with elements x, y, and z as:

(i) $(\forall x,y)\ (((x \leq y)\ \&\ (y \leq x)) \Rightarrow (x = y))$

(ii) $(\forall x,y,z)\ (((x \leq y)\ \&\ (y \leq z)) \Rightarrow (x \leq z))$

(iii) $(\forall x)\ (x \leq x)$

A linear order is any set that is partially ordered and satisfies the following:

(i) $(\forall x,y)\ ((x \leq y) \vee (y \leq x))$

If complete information was available about the relative order of each event, then the order of every pair of time points would be known and the set of time points derived from a set of interrelated events would form a linear order. However, since the knowledge available concerning the time relations of events is incomplete, the set of event endpoints forms a partial order with respect to the known instances of the relation ≤. This order can be represented by a graph, with each element of the set represented by a node, and each pair of related time points ($t_1 \leq t_2$) represented by a directed edge in the graph. Each event is represented by a pair of time points and these time points and their relations to other time points are represented in a directed acyclic graph. A labelling scheme was sought to quickly determine ordering within the graph based on relative values of the labels. Such a labelling would provide constant time determination of time point relations, and hence of the temporal relations of events.

Kameda[1975] has a constant-time algorithm for extracting ordering relationships from an acyclic digraph. However, his methods are restricted to a subclass of planar digraphs which is an unacceptable constraint for time representation. The number of labels required to permit determination of order for unrestricted acyclic digraphs is dependent on the dimension of the graph. Kameda's graphs were restricted to two dimensions and he found a two-labelling for them.

Baker et al [1980] examine partial orders of dimension two in some detail. They have shown that a two-labelling is possible if and only if the relation is a partial order of dimension less than or equal to two. In general, n-labels are required for an n-dimensional graph. So the problem becomes one of determining the dimension of the time graphs and using a corresponding number of labels. However, as Dushnik and Miller [1941] prove, for every cardinal number m there is a partial order (on some set) with dimension m. This allows the completely unacceptable possibility of requiring m-labels for every one of m nodes in the graph. As well, procedures for determining the dimension of a given partial order are acceptable only for very small sets of nodes. The problem of determining the dimension of an arbitrary acyclic digraph is difficult and may turn out to be NP-hard [Kameda, personal communication].

At any rate, these algorithms only address the problem of relative ordering. They would have to be greatly modified since dates and durations will have to be maintained as well. Aside from this, the algorithms all operate on a given graph. This excludes the possibility of inter-leaving additions to the graph with queries for information while maintaining constant-time algorithms to answer the queries.

These investigations showed that developing constant-time algorithms based on labelling the nodes of an arbitrary acyclic digraph, would require major research and might ultimately be unsucessful. Connectivity matrices,

where the relation between every pair of points is maintained in a matrix, are also unacceptable because of the requirement of $O(n^2)$ storage and $O(n)$ cost of adding information. As a result a practical approach was taken and numerous texts were analysed to try to determine some characteristics of time graphs and exploit these in question answering algorithms. The next section describes these efforts.

2.4 Characteristics of temporal knowledge

Since a constant-time labelling algorithm for unrestricted acyclic digraphs is an open problem, an effort was made to determine some characteristics of the class of "useful" time graphs. Time graphs were constructed for several newspaper stories, as well as a fairy tale (Little Red Riding Hood), and passages from a novel (Hemingway's *The Old Man And The Sea*) and from a European history text. Although these graphs displayed a high degree of variation, as expected from the wide range of literature examined, certain characteristics could be determined.

In order to demonstrate this process the time graph for the following excerpt from the fairy tale "Little Red Riding Hood" is derived. "And the wolf raced away as fast as his legs would carry him. Red Riding Hood was also in a hurry, but there was so much to look at on the way. There were birds to listen to, butterflies to chase. The wolf, in the meantime, had reached the cottage."

As a first step the text is broken into the individual events that comprise it. The events are numbered according to the sentence and clause within that sentence where they occur.

Event 16a - And the wolf raced away

Event 16b - as fast as his legs would carry him.

Event 17a - Red Riding Hood was also in a hurry,

Event 17b - but there was so much to look at on the way.

Event 18a - There were birds to listen to,

Event 18b - butterflies to chase.

Event 19 - The wolf, in the meantime, had reached the cottage.

In the following analysis of the temporal relations of these events, a time point t_i is assumed to be the last action (in this case the end of discourse between the wolf and L.R.R.H.) occurring previous to this text.

Event 16a after t_i.

Event 16b equal-to Event 16a.

Event 17a after t_i.

Event 17b equal-to Event 17a.

Event 18a during Event 17a.

Event 18b during Event 17a.

Event 19 after Event 16a & Event 19 before-end-of Event 17a.

The time graph for this example is presented in figure 1. By following this process for the entire story a time graph can be constructed as each event is encountered. Two

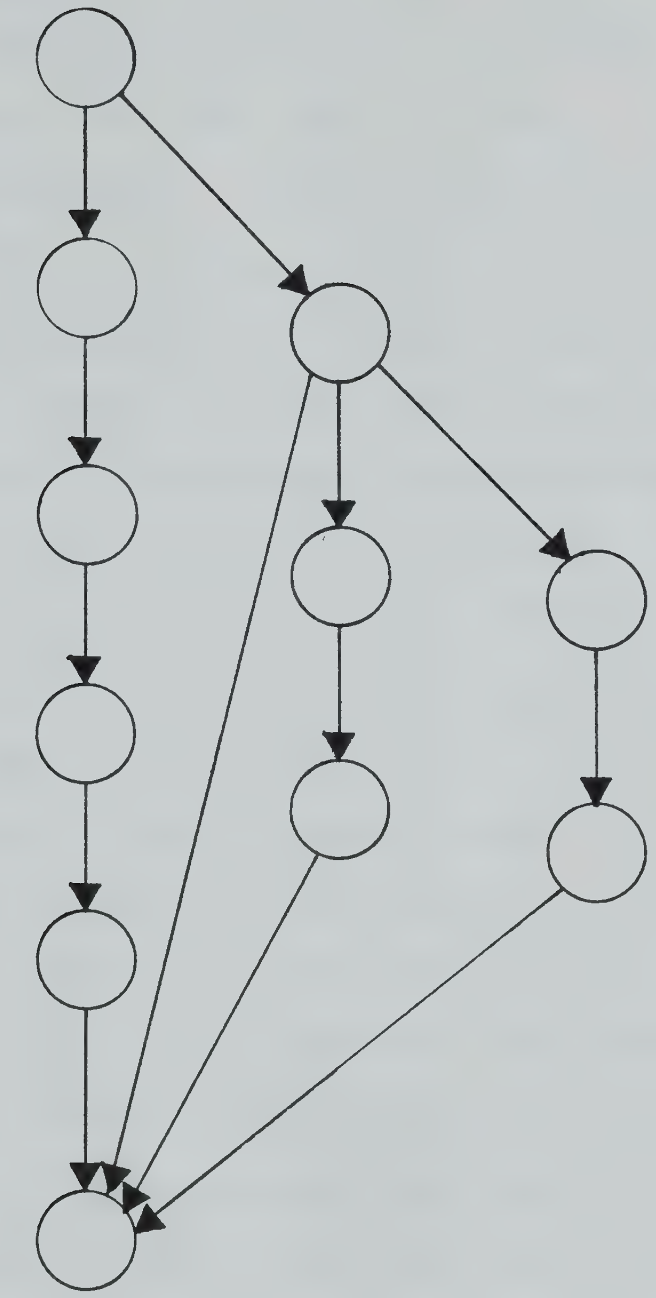

Figure 1: Time relations hand-extracted form four successive sentences in Little Red Riding Hood.

prominent characteristics of these graphs are termed the time chain and the side chain. The definitions for these terms are given in section 4.1, but a loose description can be given here. The time chain is simply a linear sequence of connected time points. Although this feature is not unexpected, it was observed that one or two extended time chains often accounted for a large proportion of the nodes in the time graph. For Little Red Riding Hood approximately seventy percent of the nodes (representing event end points) were found to be part of a single chain.

Although occurring to varying degrees in the different texts, time chains were was especially significant in the fairy tale and the novel. Narrative stories with very few or no references to specific dates showed this feature. The stories that did not follow this pattern had many more references to specific times and durations.

A side chain starts and ends with connections to a single other chain. Within the stories, this usually occurred when some previously mentioned event is later expanded in more detail. This feature is not prominent in the time graphs of any particular type of story, but common to all types.

In order to create a fast algorithm for determining the relative positioning within a time graph, a method which takes advantage of chains and side chains is needed. Clearly, for some stories, a constant-time algorithm for determination of positioning within a single chain would

include most of the time points in the story. If the algorithm is slower (but still not worse than O(n)) for time points of different chains, then it will not greatly slow the average time for question answering. Also, if the side chains could be incorporated into their superordinate chain, then the number of distinct chains in the graph would be further reduced, improving the speed further. For those graphs with chronological specifications, these could be used to speed the algorithm and compensate for the typically higher number of chains. These considerations form the basis for the temporal representation described in this thesis.

2.5 Time Frame

Although this thesis is concerned solely with the representation of time, a constant consideration is the integration of the system into the larger framework of a general knowledge system. The information provided to the time handler will be provided by the parsing and translation processes from a natural language input. An element of the parser, though not yet implemented, is the time frame.

A time frame is a temporal bracket, within which the currently discussed events are understood to take place. Natural language narratives explicitly or implicitly set up time frames, and the start and end of these time frames form upper and lower bounds on events within them. The events may in turn serve as time frames for more detailed accounts. Since the parsing and translation process will be providing

this information, the time handler must provide an easy way of specifying a time frame for an event, and an effective means of representing this information.

The graph construction algorithm (see chapter 4) will allow specification of a time frame when adding a new event to the graph. Incorporating the time frame information poses no special problems. A time frame is treated as just another event in the graph. Adding new events within existing ones results in the creation of new side chains.

As long as the representation allows events previously specified to be expanded and new events to be entered as side chains, then the representation should handle time frames effectively. Since any event is a potential time frame, the system must be able to handle any event being specified as a time frame efficiently without alterations to the representation.

3. Time Relations and Inferences

This chapter presents an overview of both the structures and methods used in assimilating temporal information, and in answering questions concerning this knowledge. The chapter emphasizes the intuitive basis of the data structures and methods. Complete, detailed descriptions of material touched on here, will be presented in chapters 4 and 5.

3.1 Organizing Time Relations

This section presents a general overview of the structures that are employed in the proposed representation. The input to this time module will come from the parsing and translation process. As translation proceeds and new events or new temporal relations between events are discovered, these are sent to the time module. As mentioned in 2.2, a representation based on time points is favoured for design reasons, and this requires relations between events to be translated into relations between the time points of the respective events.

The time points are represented as nodes in a graph, with directed arcs representing the relations between time points. A directed arc is placed between each pair of time points whose order is explicitly known. This produces an acyclic digraph[5] for any two time points t_1, t_2, $t_1 \leq t_2$ is a logical consequence of the facts represented in the graph

---

[5] This excludes time travel stories in which cycles can occur.

if and only if there is a path from t_1 to t_2 in the graph. One method (although inefficient) for deducing facts of the form $t_1 \leq t_2$, is to perform an exhaustive search for a path from t_1 to t_2. Such a search procedure must continue until either a path is found (in which case the relation $t_1 \leq t_2$ is confirmed), or until it has been determined that no such path exists (in which case a successful search for the "reverse" path, i.e. from t_2 to t_1, denies the relation $t_1 \leq t_2$, and if unsuccessful indicates the relation is unknown).

The worst case time requirements for a single search through an acyclic digraph with e edges is O(e). However, the aim of the special purpose inference mechanisms is to achieve the fastest possible question answering, and so the methods outlined here will try to capitalize on the major characteristics of the time graphs (presented in 2.4) to improve question answering speed. The representation presented here is designed to perform well for the range of graphs that will most likely be encountered.

As mentioned in 2.4, an important aspect of the graph is the time chain. A time graph represented in the manner described here, can be viewed as a collection of time chains, with directed edges connecting the collection to form a graph. This is illustrated in Figure 2.

This two level structure requires a distinction between two types of edges in the time graph. An edge connecting two nodes belonging to different chains is called a "cross-chain" link. All other edges (which will be those

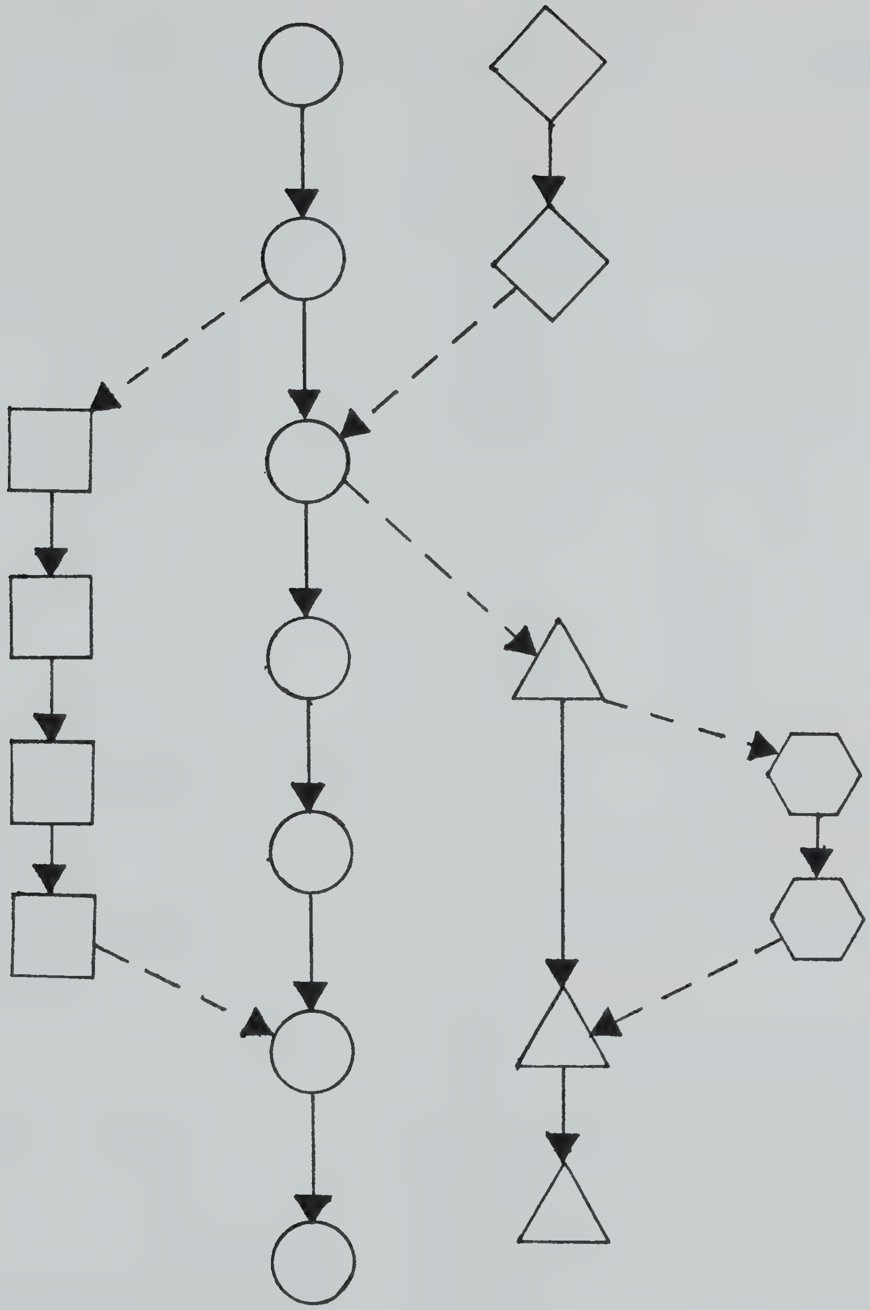

Figure 2: Time Graph Structures.
The five distinct node shapes distinguish the five time chains in the graph. Solid arrows indicate connections between nodes within the same chain, while broken arrows indicate "cross-chain links". The hexagon chain is the only side chain in this diagram, the square chain being excluded by the full definition given in chapter 4.

connecting nodes of the same chain), are referred to as 'chain' links. A distinction is not made in the representation of the two edges in the time graph, but in the processing of the two types.

The nodes within a single chain form a linear order. Since only a single label is necessary to distinguish (in constant time) relative position within a linear order (Baker et al [1980]), a "pseudotime" is assigned to each node. The time graph consists of a collection of time chains, each with its' own pseudotime sequence. The pseudotimes permit constant time determination of relative position within a chain, by a single comparison of pseudotimes. This solves part of the representation requirements, but still leaves the question of accomodating inter-chain connections without severely hampering efficiency.

In order to deal with inter-chain connections, a "meta-time-graph" is constructed. This higher level graph reduces each chain of the time graph to a single node, removing all chain links, but retaining all cross-chain links. The position within each chain where the connections occur is recorded on each edge in the metagraph. The methods for inferring relations between nodes of different chains are discussed in the next section, but they involve operations on the metagraph, not the time graph. Any operations on the metagraph will be much faster than the time graph since it is expected, from the temporal analysis

of numerous stories, that there will be far fewer chains in the graph than nodes.

This completes the foundation for the representation scheme, but it allows for only one type of temporal knowledge. Aside from time order, absolute times and durations should be representable. Therefore, the representation provides for the specification of an absolute time for every time point. Whatever portions of a date are specified or can be determined from contextual or world knowledge, are included. Of course, the degree of precision in the specification of a date is story-dependent. For example, days of the week may be regularly specified in some story, while the calendar date is never mentioned. In order to provide a high degree of flexibility, a date is broken into a pair, representing the best lower and upper bound that can be derived for that date.

For example, suppose there is an event representing "John ate dinner", with a starting date determined as July 10, 1983 in the evening. First a bound for "evening" must be obtained from world knowledge. If it is assumed that "evening" normally starts and ends at 1800 and 2200 hours respectively, then a lower bound for this event can be determined as 1800 hours on July 10, 1983. Assume that it has already been determined from contextual information that it took between one and two hours for John to finish dinner.

The event consists of two time points, t_1 and t_2, representing respectively, the start and end of the event.

The bounds on t_1, will be the earliest and latest times that the event could have started. So the lower bound for t_1 will be July 10, 1983 at 1800 hours, and the upper bound July 10 at 2200 hours. The bounds on t_2 will be the earliest and latest times that the event could have ended. So the lower bound for t_2 will be 1900 hours, and the upper bound 2400 hours, both on July 10, 1983. While this is an unusual example, in that the bounds on each of the two time points could be easily determined, it serves to illustrate how the bounds are set on the time points of each event.

As with dates or absolute time specification, each duration is a pair representing the lower and upper bound of the duration. Durations can occur between events, or they may be the duration of one or more events themselves. If the duration is between events, then the lower and upper bounds are placed on the edge of the graph which connects the two events. If the duration is of an event, then the bounds are placed on the edge which connects the two time points composing the event. Thus, in the above example, the edge connecting the beginning and end of John's dinner would be labelled with lower bound one hour and upper bound two hours. All of the above information concerning John's dinner is presented in Figure 3.

In this way, all durations are broken into lower and upper bounds and placed on the appropriate edges in the time graph. All absolute times are also broken into bounds and placed on the appropriate nodes in the time graph. If a

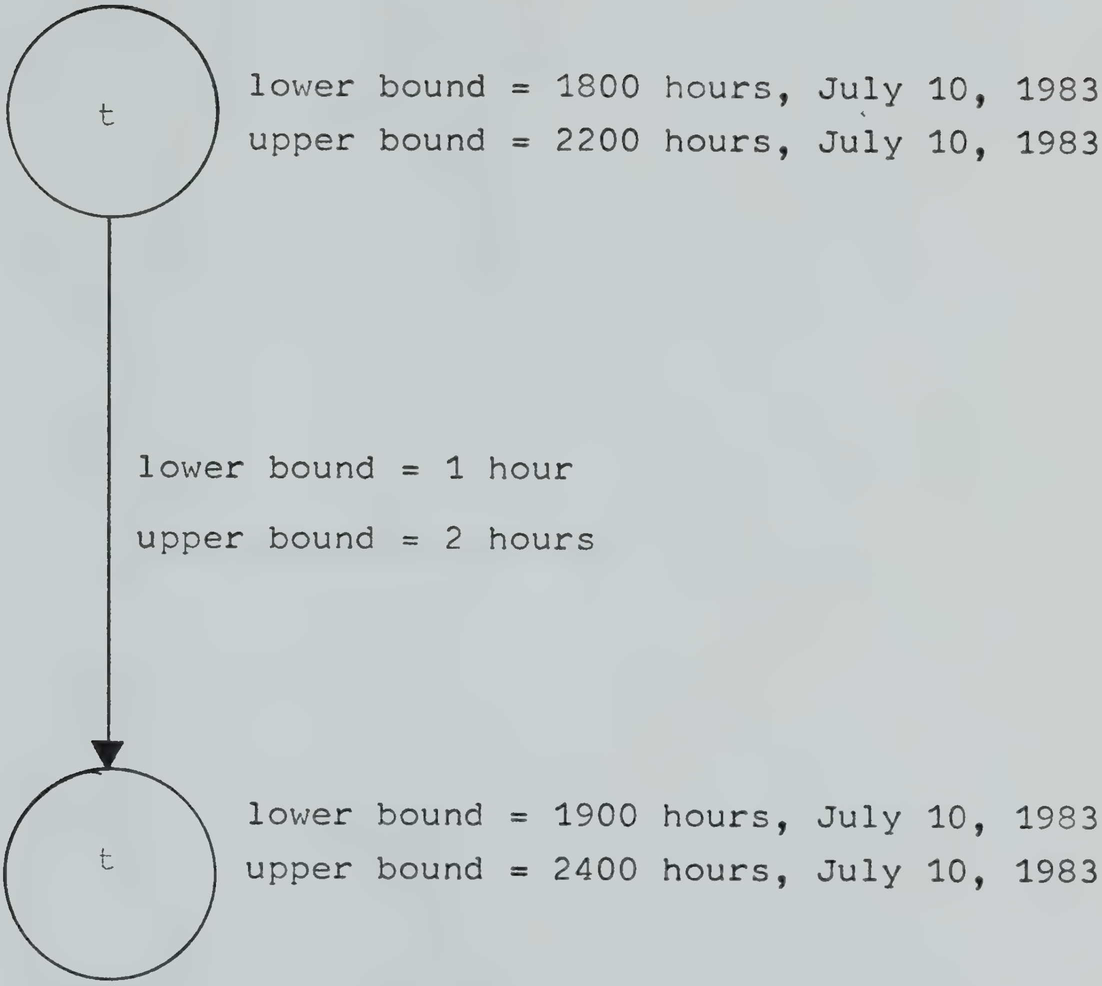

Figure 3: Absolute Times and Durations for "John's Dinner".

The top node (t_1) represents the start of the event and the bottom node (T_2) represents the end of the evnt. Absolute time bounds are indicated beside the node and duration is indicated beside the edge.

duration is specified between two nodes in the graph not directly connected by an edge, then a new edge may be created between them[6] to hold the new durational information. This assumes that for every duration specified, the time order is known, and if this is not the case the system will reject the duration.

These are the basic forms of the representational structure of the time handling system. There are some important activities that occur during graph construction, such as propagation of absolute times, that have not yet been discussed. These, along with more detailed and precise descriptions of the representation are left to chapter four.

3.2 Retrieving Time Relations

As discussed previously, there are two labels on each node of the graph which are used exclusively for question answering. One label indicates the chain to which that node belongs, and the second (the pseudotime) represents the relative position within the chain that the node occupies. If two nodes are in the same chain, then a single comparison of the pseudotimes is sufficient to determine the relative order of the two nodes. If the chains are distinct, then to find the relative order of two nodes t_1 and t_2, a path must be found from t_1 to t_2, or from t_2 to t_1. The node at the start of the path if one exists[7] represents the earlier time

---

[6] Providing the new edge does not create a cycle, i.e. contradict time order information already contained within the graph.

[7] Both paths cannot exist since the graph is acyclic.

point.

The metagraph is used to hasten the discovery of a path between nodes of different chains. In rough terms, the algorithm starts at a node in the metagraph representing one of the chains of the two nodes and searches the metagraph from there, for a node representing the chain of the other node. If successful and the nodes are also on the path within their respective chains, then the node that starts the path is before the node at the tail. If unsuccessful, then it is necessary to repeat the process starting from the second node and looking via the metagraph for a path to the first. If this is successful then the first node is after the second. If this second attempt is also unsuccessful, then the relative position cannot be inferred from the time graph.

Examination of directed paths in the time graph is one method of determining relative order of time points. A second method is to examine the absolute times[8] of the time points in question. It is quite possible, especially in stories dealing with many absolute times, that comparison of the upper and lower bounds on a pair of nodes will determine their relative position. Since this involves few (at most four) comparisons it can usually be done faster than a search of the metagraph, and for this reason is attempted first in order to get a very quick response to a question.

---

[8] These may not be given explicitly, but the propogation of absolute time bounds through the graph results in a high proportion of nodes with absolute time bounds.

Of course if this is successful, then a more costly search of the metagraph can be avoided.

The relative order of time points and events is only one type of information that needs to be extracted from the graph. It is also desirable to be able to retrieve the best time bounds for any time point or event. This includes bounds which are improved as a result of inferences from relations to other time points and absolute times within the time graph. The main objective of this time representation is to match the inference capabilities that humans can exhibit in the time domain. Human cognitive skills are usually much weaker in dealing with specific dates, as compared to time order. However, if this representation can be extended with little effort to handle such inferences efficiently, then the extension will be worthwhile.

In order to do this, extra computations are performed when new information is added to the graph. A propagation algorithm is proposed to spread improved time bounds through the graph so that every time bound is the best that can be derived from direct and inferred time knowledge.

The propogation techniques are described more fully in chapter 4. Basically, absolute times are propagated through the graph, so that at every node the lower bound is equal to the greatest lower bound of all ancestor nodes in the graph, and the upper bound is the smallest upper bound of all descendant nodes. By maintaining the absolute times of the graph in this fashion, a lot of computational effort is

shifted from the question answering algorithms to the graph construction algorithms. This results in fast determination of absolute time bounds for any given node. The time bounds are simply extracted from the particular node.

Durations, although represented in basically the same manner as absolute times, are treated somewhat differently. Durations are used to update the absolute time bounds on the nodes directly associated with the duration. Durational information is not updated in the same manner, however. New information added to the graph that improves one or more absolute times is not used to update durations.

To extract durations, the question answering algorithms attempt to obtain the duration from a path between the relevant nodes. If there exists explicit durational information on the edges in the graph, then this is used. If a duration is requested between two nodes, and no edge exists between these two nodes with either a lower or upper bound duration on it, then the algorithms attempt to derive the missing information dates on these nodes. By subtracting the lower and upper bounds of the absolute times on two nodes, a lower and upper bound on the duration between the two nodes can be derived.

From this description of the inference and retrieval mechanisms, it should be clear that with the exception of relative order of nodes of different chains, question answering takes place in constant time. A fuller and more detailed description of the algorithms and the pre and

post-conditions for which they are defined is given in chapter five, along with the complexity analysis.

4. Graph Construction

This chapter focuses on the details of the time graph and its construction. The first section examines data structures that are necessary for the basic time graph representation. Also included is a description of some of the supporting structures used to speed up the construction and question answering algorithms. The representation for absolute times and the advantages over previous representations are discussed. The second section outlines the graph construction algorithms which maintain the structures outlined in the first section. Finally, the time and space complexity for the graph structures and for the graph construction algorithms are examined.

4.1 Time Graph Structures

This section deals with the internal organization of all forms of temporal information in the time module. The first part describes the temporal graph itself, and provides definitions to be used throughout the remainder of this thesis. The second part discusses the details and possible extensions of the meta-graph, while the last part deals with the representation of absolute times.

4.1.1 Elements and Data Structures of the TimeGraph

Every event is viewed as two time points representing the start and end of the event. We use a representation similar to Hirschman's [1981] in that each time point is

represented as a node in a directed graph. An edge of the graph represents the before-after relation between the time points it connects. The following definitions are used throughout the remainder of this thesis.

A time graph G = (T, E) is an acyclic directed graph in which T is the set of vertices (nodes) and e is the set of edges(links). The information stored in the graph is usually in the form of pairs of time points representing events. (Although in some cases there may be single time points which do not represent event boundaries.) The relation '≤' forms a partial order over the time points and each directed edge (t_i, t_j) in the graph represents the temporal relation $t_i \le t_j$.

Assume there are n nodes, and e edges in the graph, represented by ordered pairs of nodes (eg. (t_1,t_2)). A node t_i is a descendant of a node t_j (and t_j is an ancestor of t_i), if and only if there exists a path from t_j to t_i. A node t_i is a direct descendant of t_j (and t_j is a direct ancestor of ti), if and only if there exists an edge (t_j, t_i) in the graph.

A chain is any one of a set of linear paths into which G is algorithmically partitioned. Each linear path consists of a series of nodes t_1, t_2, ... , t_m ($m \ge 1$) such that for all i ($1 \le i \le m$), t_i is a direct descendant of t(i-1). The partitioning ensures that every node belongs to one and only one chain. In the implementation each chain has a unique number (pseudotime) associated with every node of that

chain, identifying the chain to which it belongs.

Let K represent the chain complexity, defined as the number of chains in the graph as determined by the graph construction algorithms. For any given graph G, there may be more than one possible value for K. The calculation of K is dependent upon the order in which the edges and nodes are added to G.

When a node is added, K may increase. In most cases the node will have a known time order relative to one or two existing nodes and hence at least one new edge will be created. The effect on K is then given by the following rules, where t_j is the new node and t_1 and t_2 are nodes already in the graph:

```
 (1) Suppose Edge($t_j$,$t_2$) is added;
    if $t_2$ is the first node of a chain, then $t_j$ adopts this type,
    else $t_j$ is assigned a new chain and ($K \longleftarrow K + 1$).
 (2) Suppose Edge($t_1$,$t_j$) is added;
    if $t_1$ is the last node of a chain then $t_j$ adopts this type,
    else $t_j$ is assigned a new chain and ($K \longleftarrow K + 1$)
 (3) Suppose Edge($t_1$,$t_j$) plus Edge($t_j$,$t_2$) are added;
    if $t_1$ and $t_2$ are of the same chain then
        if there are no other nodes between $t_1$ and $t_2$ then
        $t_j$ adopts this chain and will be inserted between $t_1$ and $t_2$,
        else $t_j$ is assigned a new chain and ($K \longleftarrow K + 1$)
```

else an edge(t_1, t_j) is added as in (2) above and then edge(t_j,t_2) is added.

If the rare situation occurs where a new node is added to the graph without any connecting edge, then this node creates a new chain (K <--- K + 1).

We make a distinction between two different (logically, but not physically) types of edges. When a new edge (t_i, t_j) is added to G, where no such edge previously existed, and t_i, t_j are nodes of G within distinct chains, then (t_i, t_j) is called a "cross-chain link". All other edges in the graph are called "chain edges", as they connect two nodes of the same chain.

If m is the number of cross-chain links, and p the number of chain links in the graph, then **$E = p + m$**.

As with K, the value of m is dependent on the order of introduction of the edges of the graph. The following example illustrates this point.

Consider nodes (t_1, t_2, t_3, t_4, t_5). Suppose that edges are introduced in the following order: <(t_1, t_2), (t_2, t_3), (t_2, t_4), (t_4, t_5)>. Now, a new edge(t_1, t_4) is a link between two different chains and hence increments m. But if the same edges are introduced in the order: <(t_1, t_2), (t_2, t_4), (t_4, t_5), (t_2, t_3)> then the edge(t_1, t_4) is between nodes of the same chain, and hence does not affect the value of m. This is demonstrated in Figure 4, in both cases K = 2.

If a node t_j is a descendant but not a direct descendant of node t_i, then an added edge (t_i, t_j) is a

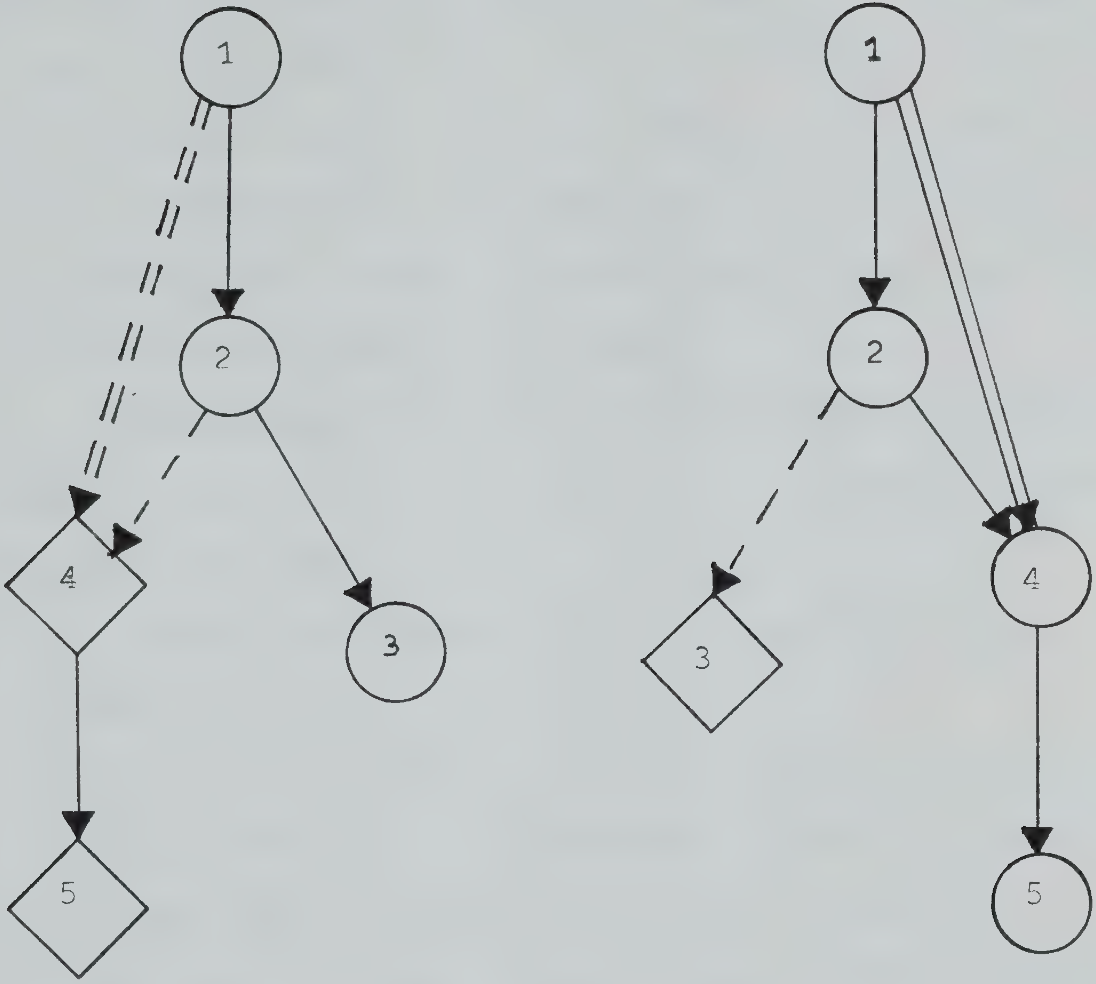

Figure 4: Different Time Graphs for the Same Information
The two figures show two different time graphs resulting from the introduction of identical time ordering information, but introduced in different order. Adding (t_1, t_4) can add a cross-chain link and increase m (left) or add a chain link and leave m unchanged (right), depending on how the graph construction algorithms partition the graph.

transitive edge adding no new information to the graph. In general no effort is made to exclude transitive edges from the graph. However, a transitive edge (t_i, t_j) where t_i and t_j are of the same chain, will not be added unless there is new information which must be added on that edge[9].

A pseudotime has already been described in 3.1, and is a unique number (within a single chain) which delineates the time order of a particular chain.

A side chain of chain "A" is defined as a chain with head(th) and tail(tl), connected to the nodes ta, tb of chain A, such that;

1) There exist edges, (ta, th) and (tl, tb) and (ta, tb).

2) For all edges (ta, tj), ta.chain = tj.chain ⇒ (tb.pseudotime < tj.pseudotime) where $(1 \le j \le n)$

An example of a side chain is given in Figure 1.

An array of time points is used to maintain a direct link to any given node in the graph. Each entry in the time point array contains a pointer to the node in the time graph which represents this point. It is not necessary to anticipate the maximum number of time points to ever be incorporated in the graph. If the array is allocated some reasonable amount of storage initially and subsequently runs out, it would be possible, since the time handling mechanism is not designed to be an independent module, to have an external program allocate a new array and link it to the old

---

[9] Specifically, a new duration which must be maintained on that edge.

one.

Each node in the graph is dynamically allocated whenever a representation for a new time point is required. Each node in the graph contains the following fields:

1) A time point number. This is the distinct external number by which the time point is known in the system.

2) A chain number. This identifies the chain to which the node belongs.

3) A pseudotime. This is the single-labelling scheme which orders all the nodes within a chain and provides constant time determination of relative order in each chain.

4) A descendant list. This is a linked list of pointers to direct descendants in the graph.

5) An ancestor list. This is a linked list of pointers to direct ancestors in the graph. Although the descendant list contains the edges required to maintain all the information discussed above, the ancestor list is necessary to reduce processing during addition of new absolute times. The fields for storing absolute time information are described in section 4.1.3.

Aside from the time graph itself, there are supporting structures which are crucial to the question answering and graph construction techniques. The first of these is a pair of arrays which maintain additional information about each chain in the graph. The maximum and minimum pseudotime of each chain in the graph is kept separately in these two arrays. This is information which could be obtained from the

graph without the separate structures, but only at the expense of searching chains. The purpose of the maximum and minimum values will be discussed in the section on algorithms for graph construction 4.2.1.

4.1.2 The Metagraph

The meta-graph is used to speed the process of searching the time graph for paths between two nodes of different chains. For each chain in the graph an entry representing that type is maintained in the metagraph. Each chain A is reduced to a single entry in the metagraph, plus a linked-list of other directly connected chains.

If there is a direct connection between chain A and chain B in the time graph, then there must exist a pair of nodes (t_i and t_j) in the time graph such that; t_i belongs to chain A, t_j belongs to chain B, and t_j is a direct descendant of t_i. This means that metagraph edges correspond one-to-one to cross-chain links in the time graph. If a cross-chain link E_i connects a node of chain A to a node of chain C, then the connection is entered under the chain connection list of chain A in the metagraph.

The existence of a connection between two chains does not guarantee that any pair of nodes in the two respective chains can be ordered. The ordering relationships depend on location within the chains where the connection occurs. Hence each edge in the metagraph records this information as well.

So every chain in the graph has an entry in the meta-graph containing a list of directly connected chains. Every element of the list contains the particular chain connected, as well as the pseudotime of the ancestor node in the connection and the pseudotime of the descendant node in the connection. This information is sufficient for efficient recovery of ordering information implicit in any pair of connected chains.

An example of how a time graph might be broken up into chains, which form the nodes of the metagraph is given in Figure 5 and Figure 6. The chains of the time graph are circled to indicate the reduction that will take place in the metagraph. Since there are typically a large number of nodes in one or two chains, this reduction can greatly reduce the number of nodes and of course completely eliminates all chain links.

4.1.3 Absolute Times and Durations

In addition to the structures described in the previous sections which are sufficient for time order inference, there are further structures needed to handle absolute (or chronological) times and durations. Each node of the graph contains the fields described earlier, plus two bounds which are used to record absolute time information if it is present. Similarly, each edge of the time graph, in addition to a pointer, contains a lower and upper bound to record information about the duration between the two nodes

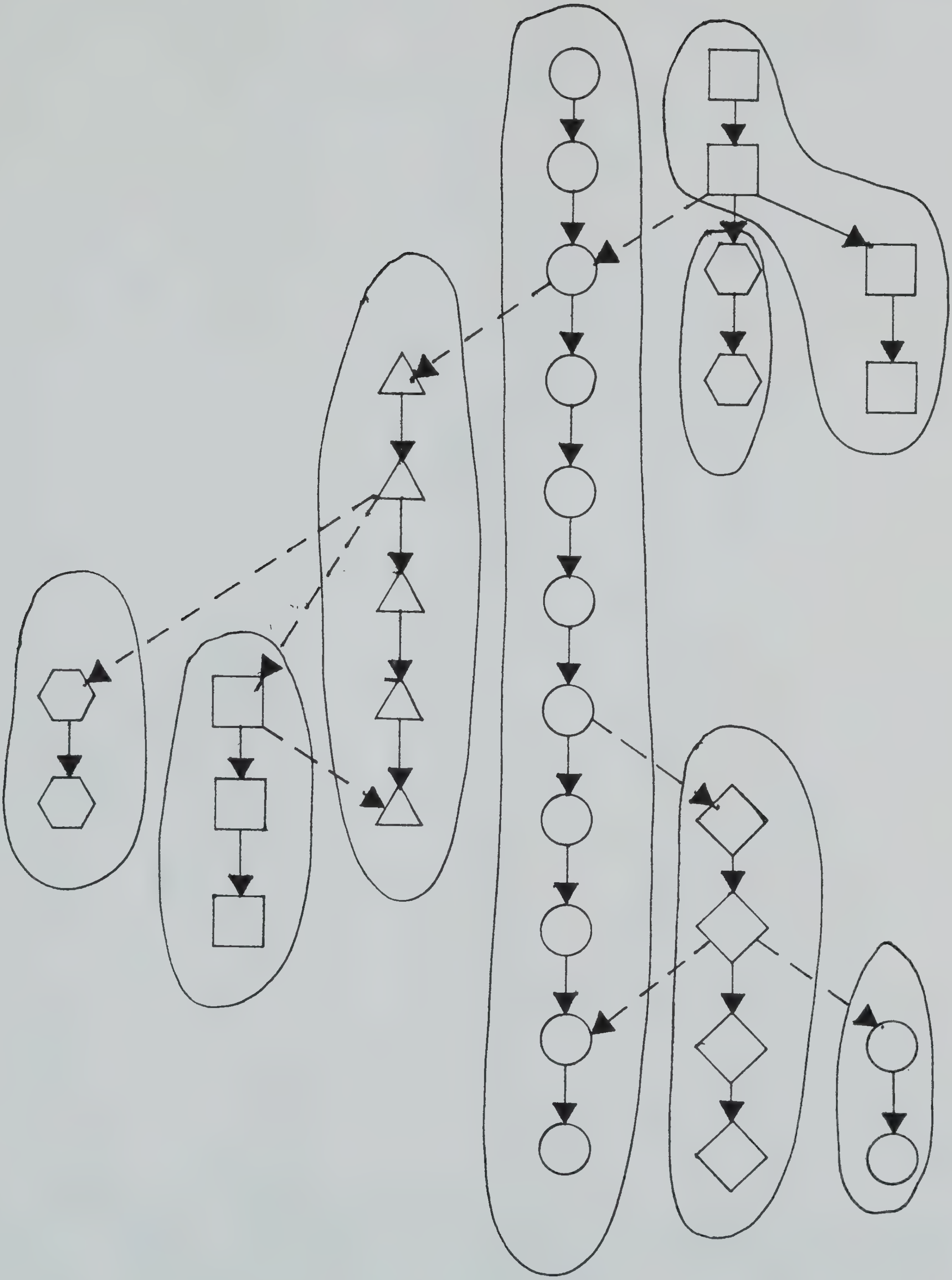

Figure 5: A Time Graph G_1 Partitioned into Chains. The nodes of each chain have distinct shapes from their neighbours.

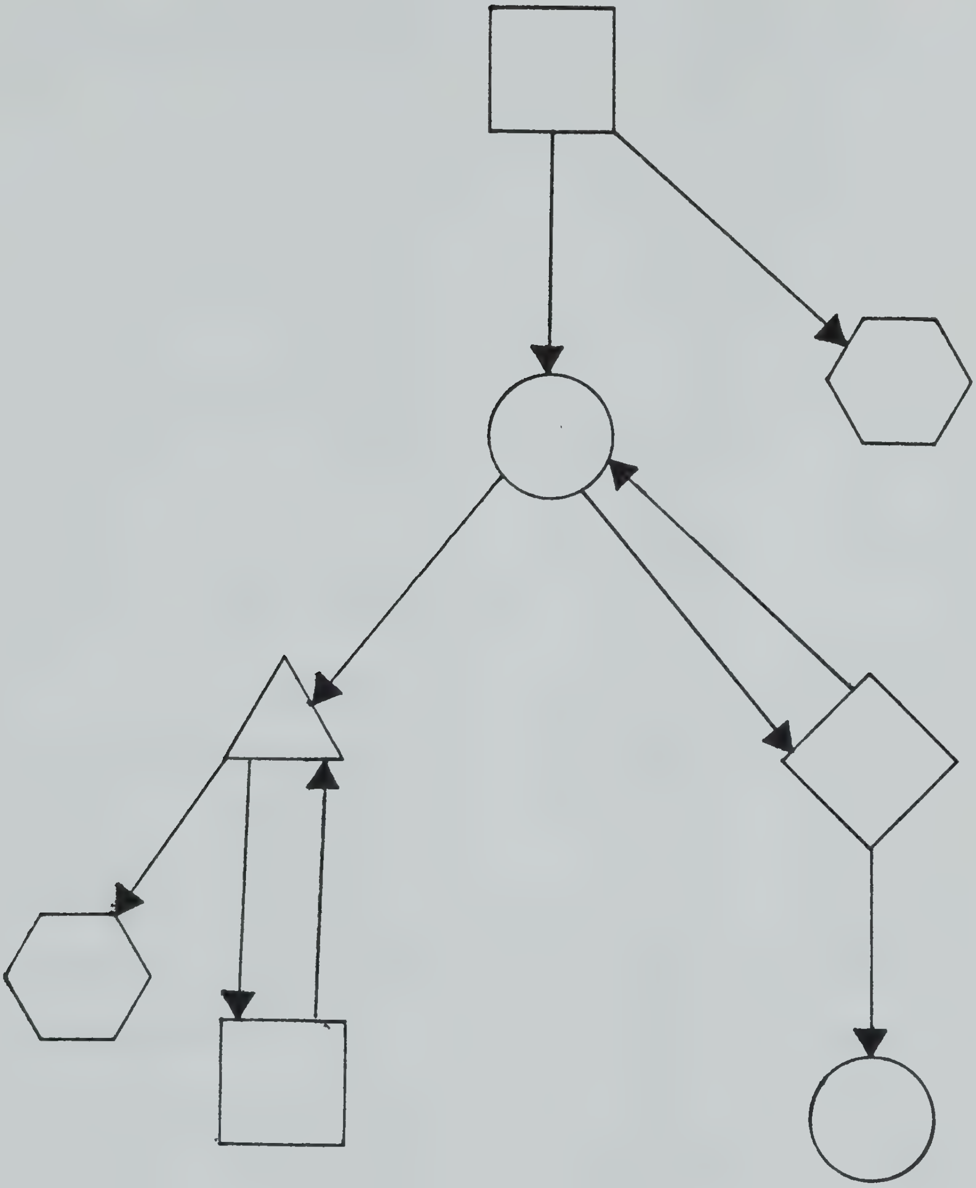

Figure 6: The Metagraph for G_1.
Each node represents a chain (corresponding to the shape) in the time graph G_1. Each cross-chain link in the time graph is included in the metagraph.

connected by that edge.

Virtually every system that deals with temporal knowledge deals with the issue of recording absolute time information. Previous work in contending with this information provides two key insights. First, the representation must allow a high degree of flexibility in the specification of an absolute time. This is because absolute times and durations can be specified more or less precisely, depending on the smallest units used as well as other factors (such as whether exact or vague numerals are used in time adverbials).

This type of imprecision has typically been handled by previous researchers in temporal knowledge by employing fuzzy expressions and arithmetic (Kahn and Cohen). Such a method involves pre-determining an expected time and the amount of fuzziness associated with each time adverbial. Examples of such adverbials are, "a week ago", "a few years later" and "in a couple of days". In order to add or subtract dates with varying degrees of fuzziness, these authors design fuzzy arithmetic modules.

This approach appears to be excessively complicated as far as representation is concerned and over-simplified as far as natural labguage interpretation is concerned. The correct interpretation of vague expressions can be highly context dependent. A sentence such as, "I have not seen John for a while", could express a duration of years or hours depending on context. Since the problem is not restricted to

temporal knowledge and dealing with it accurately involves incorporating knowledge from other areas, interpreting vagueness should be a part of general language understanding and not a component of the time module. To a first approximation, at least, vaguely specified times and durations can be effectively represented by using upper and lower bounds as proposed in this thesis.

An absolute time in this representation consists of six fields representing year, month, day, hour, minute and seconds. Each of the first five of these fields will accept both numerical and alphabetic characters, while the seconds field is a real number permitting as many significant digits as desired. Also any or all of the fields may be specified as unknown, which is a state distinct from all others, avoiding any ambiguity in the representation.

The following examples illustrate how time specifications are represented. For purposes of these examples a "?" is used to represent an unknown quantity, l and u represent the lower and upper bounds assigned to an event, and spaces separate fields of the absolute times.

1) "John saw Mary a few minutes before noon on September 6, 1982."

l = "1982 10 06 11 50 0.0"

u = "1982 10 06 11 59 0.0"

2) "In July I broke my leg."

l = " ? 07 01 00 00 0.0"

u = " ? 08 01 00 00 0.0"

Relying solely on numeric dates such as these, causes a representation problem when dates or durations are related, but unknown. This occurs frequently in natural language discourse, when mention of one event is made and later, some event is mentioned as occurring in the same year (season, month, etc.) although no specific time is mentioned for either event. To handle this situation a representation is desired which will maintain the relation between events so mentioned, and not make invalid inferences regarding the relation of these events to the rest of the events in the time graph.

Consider a sentence following (2) in the above example, "That was the same summer I quit smoking." Now, although the year referred to remains unknown, some means of indicating that it is the same year as in (2) is necessary. To handle situations such as these, alphabetic characters representing constant (but unknown) values are used. In this example a constant "t" might be assigned to the year of both absolute time specifications. This would produce:

2a) "In July I broke my leg."

l = " t 07 01 00 00 0.0"

u = " t 08 01 00 00 0.0"

3) "That was the same summer I quit smoking."

l = " t 06 01 00 0.0"

u = " t 09 01 00 00 0.0"

(This is the time frame within which "I quit smoking" occurs.)

This also allows relative durations to be indirectly represented. By assigning one duration $(t_2 - t_1)$ the value "a", and another $(t_4 - t_3)$ the value "b", relations between the two durations can be expressed in terms of a and b without knowing the actual duration. So $(t_4 - t_3) = 2(t_2 - t_1)$ becomes $b = 2a$. The relations between these constants are represented at a higher level of the knowledge system.

Previous time representations have either filled in missing information with defaults, with guesses from context or other special reference events, or left these portions unknown. The interval techniques presented here are both simple and provide a means (although indirectly) of representing relations between unknown absolute times and durations.

Allowing an absolute time to contain a mixture of constants and numeric values causes some comparison problems. To compare two absolute times l_1 and l_2 the algorithm starts with the most significant field (the year) and progresses to less significant fields (month, day, minute, second) until either the larger time has been determined, two fields are found incomparable, or all fields have been examined (in which case since neither has been determined larger, the two must be equal). Two fields of an absolute time are comparable if and only if in each field there exists either positive integer values or identical constants. This means that different constants, or constants and values in the same field cannot be compared, and hence

if the times are identical up to this field, the two times are incomparable.

4.2 Algorithms for Graph Construction

This section provides a detailed description of the graph construction algorithms. The discussion is broken into two parts, the first dealing with the algorithms for adding new events, time points and relations between these. The second part describes the algorithms for adding new absolute times or durations, and describes how new inferred times can be calculated and propagated through the graph.

4.2.1 New Events and Relations

Aside from concerns of efficiency and completeness there is another important factor that influences the design of the algorithms. Examination of the complexity of the question answering algorithm (section 5.3) shows that the worst case for the algorithm is directly proportional to the values of K (chain complexity) and m (number of cross-chain links). The average complexity appears to be proportional to K and m as well, as can be observed from the test results given in section 5.5. Since the primary concern of this research was to find fast retrieval mechanisms, minimizing the values of K and m became objectives of the graph construction algorithms.

Consider the addition of a single time point t_i related to some other time point(s) t_j (and possibly t_k) already in

the time graph. The most common relations that can exist between time points are the following:

1). t_i AFTER t_j

2). t_i BEFORE t_j

3). t_i DURING t_j t_k

4). t_i EQUAL t_j

A sketch of the algorithms for each of the four relations in this case is given below.

```
 1) $t_i$ After $t_j$
if $t_j$ is the last node in a chain then
    $t_i$.chain := $t_j$.chain
    $t_i$.pseudotime := $t_j$.pseudotime + increment
    add a chain link, $t_j$ to $t_i$
otherwise
    $t_i$.chain := NewChain
    $t_i$.pseudotime := 1
    add a cross-chain link, $t_j$ to $t_i$


 2) $t_i$ Before $t_j$
if $t_j$ is the first node in a chain then
    $t_i$.chain := $t_j$.chain
    $t_i$.pseudotime := $t_j$.pseudotime - increment
    add a chain link, $t_i$ to $t_j$
otherwise
    $t_i$.chain := NewChain
    $t_i$.pseudotime := 1
    add a cross-chain link, $t_i$ to $t_j$
```

```
 3) $t_i$ During $t_j$ $t_k$
if $t_j$.chain = $t_k$.chain then
    $t_i$.chain := $t_j$.chain
    add chain link, $t_j$ to $t_i$
    add chain link, $t_i$ to $t_k$
    if there is room in the interval between
    $t_j$.pseudotime and $t_k$.pseudotime then
        $t_i$.pseudotime := a pseudotime between
        $t_j$.pseudotime and $t_k$.pseudotime
    otherwise
        renumber the chain
otherwise
    if $t_j$ is the last node in a chain then
        $t_i$.chain := $t_j$.chain
        $t_i$.pseudotime := $t_j$.pseudotime + increment
        add chain link, $t_j$ to $t_i$
        add cross-chain link, $t_i$ to $t_k$
    otherwise
        if $t_k$ is the first node in a chain then
            $t_i$.chain := $t_k$.chain
            $t_i$.pseudotime := $t_k$.pseudotime - increment
            add chain link, $t_i$ to $t_k$
            add cross-chain link, $t_j$ to $t_i$
        otherwise if all of these conditions fail then
            $t_i$.chain := NewChain
            $t_i$.pseudotime := 1
            add cross-chain link, $t_j$ to $t_i$
```

add cross-chain link, t_i to t_k

4)t_i Equal t_j

In this case the array entry for t_i is set to point to the node for t_j (which is indicated by an array entry for t_j)

The relation During must be expressed with the two time points t_j and t_k already existing in the graph. The other relations may be specified with neither or both of the time points involved being new time points in the graph. If both time points are new, then one of them is created starting a new chain, and the other is then added to the graph according to one of the above methods. If both already exist in the graph then all that needs to be done is to add the appropriate edges.

An event is the normal input element resulting from the interpretation of natural language texts. The addition of new events is dealt with in terms of time points. But before discussing the addition of new events and relations between them, examination of accomodation of time frames is necessary.

Typically, time frames in narratives correspond to previously mentioned events or states of affairs which are expanded into constituent events later in the narrative. For example, "driving to work" might constitute an event in some story, which would be translated and represented in the time

graph as soon as it was encountered in the story. Later however, more events might be mentioned, filling in "the drive to work". For example, "driving over the bridge" and "running a red light", might later unfold as details. If the translation process is working correctly these should be recognized as taking place within the time frame "driving to work". This whole frame might itself be incorporated within the description of some larger event, perhaps "the day I was fired". Similarly, each of the events might later represent time frames for other descriptions, "I saw the ferry leaving" might be an event occurring within the time frame of "driving over the bridge". In view of this, event relations were expanded to make specification of some event as the time frame for a new event a relatively simple operation.

The following outline of the algorithms for adding new events to the graph assumes that E_i is a new event and that E_j and E_k are events already existing in the time graph. The most common[10] relations for including a new event are:

1) E_i Equal E_j

2) E_i During E_j E_k

3) E_i After E_j E_k

4) E_i Before E_j E_k

The third event specified in relations three and four is optional and if included, indicates the time frame that

---

[10] There are other less common relations that may occur between events, but these can be expressed in terms of one or more time point relations, except for relations mentioned in 1.2.

the new event E_i is within. The third event is also optional for the relation During. If present then E_i is between the two events E_j and E_k, otherwise E_i is assumed to be during the event E_j. The time points for each event are indicated by E_i.st and E_i.end, representing the start and end points respectively of the event E_i. All of the event relations are translated into one or more time point relations, using the start and end points of the event.

```
 1: $E_i$ Equal $E_j$
set time point $E_i$.st Equal $E_j$.st
set time point $E_i$.end Equal $E_j$.end


    2: $E_i$ During $E_j$ $E_k$
if $E_k$ is present then
    set time point $E_i$.st During $E_j$.end $E_k$.st
    set time point $E_i$.end During $E_i$.st $E_k$.st
otherwise
    set time point $E_i$.st During $E_j$.st $E_j$.end
    set time point $E_i$.end During $E_i$.st $E_j$.end


 3: $E_i$ After $E_j$ $E_k$
if $E_k$ is present then
    set time point $E_i$.st During $E_j$.end $E_k$.end
    set time point $E_i$.end During $E_i$.st $E_k$.end
otherwise
    set time point $E_i$.st After $E_j$.end
    set time point $E_i$.end After $E_i$.st
```

```
 4: $E_i$ Before $E_j$ $E_k$
if $E_k$ is present then
    set time point $E_i$.end During $E_k$.st $E_j$.st
    set time point $E_i$.st During $E_k$.st $E_i$.end
otherwise
    set time point $E_i$.end Before $E_j$.st
    set time point $E_i$.st Before $E_i$.end
```

The time point relations are called in an order that minimizes the number of new chains that will have to be created, thus aiding question answering. However, new chains and cross-chain links will inevitably be introduced, and when this occurs the meta-graph must be updated to reflect the new time graph. Every time a new chain is created indicated in the algorithms by an assignment of the form ".chain := NewChain", a new node is created in the meta-graph. Every new cross-chain link adds a new edge to the meta-graph.

A new time point t_i entered as During t_j t_k can cause a renumbering of a chain. This occurs when t_j and t_k are in the same chain. In this case t_i should also be entered in this chain and should receive a pseudotime between those of t_j and t_k. However, it is possible that because of repeated insertions there is no longer a large enough difference between the two values of t_j and t_k to assign a new value. When this occurs it is necessary to renumber the pseudotimes

of every node in that chain. This requires starting at the first node in the chain and following the chain through the graph, reassigning the pseudotimes of each node. Also, every entry in the metagraph that connects a node of the renumbered chain must be altered to reflect the new pseudotime values as well. While this is a relatively simple procedure it may involve a large number of nodes if the chain is one of the major chains of the time graph.

This completes the description of the algorithms for adding new events and relations to the time graph. The complexity of these algorithms is discussed in the final section of this chapter. The next section describes the algorithms for adding new absolute times and durations to the graph.

4.2.2 Addition of Absolute Times and Durations

Aside from new events and relations, new absolute times and durations can also be added to the graph at any time. While this does not involve the creation of new structures it does involve much more consistency checking and computation than the additions to the graph considered above.

First consider the case of adding a new time bound on a node t_i. Assume the lower and upper bounds of any time point t_i are referred to as t_i.lower and t_i.upper respectively. If the new bound is a lower bound (NewLower) then the following consistency checks are made to ensure that the bounds stored

are the best possible bounds at all times:

```
    1):if there exists $t_i$.lower then
        NewLower > $t_i$.lower
    2):if there exists $t_i$.upper then
        NewLower < $t_i$.upper
```

Similarly, for a new upper bound on t_i (NewUpper):

```
    1):if there exists $t_i$.upper then
        NewUpper < $t_i$.upper
    2): if there exists $t_i$.lower then
        NewUpper > $t_i$.lower
```

Both of the conditions must be satisfied or the new bound is not entered. It is unnecessary for the consistency checks to examine other nodes in the graph because of the propagation of absolute times discussed below. These propagations ensure that there will be no ancestor of t_i in the graph with a lower bound greater than the one at t_i. Similarly, there can be no upper bound on a descendant of t_i with an upper bound less than the one at t_i.

Since edges in the time graph represent the ordering relation '≤', it follows that for any given node t_j, all nodes t_k such that t_k is a descendant of t_j must not occur earlier in time than t_j. Therefore any lower bound on t_k (t_k.lower) must be greater than or equal to a lower bound on t_j. So an improved (i.e. greater) lower bound on t_j, implies that on all descendants t_k.lower may also be improved to at least this new lower bound on t_j. Similarly, an improved (i.e. lower) upper bound on t_j (t_j.upper), implies that on

all ancestors t_i of t_j, t_i.upper may also be improved to at least this new upper bound.

Following these rules every time a new time bound is entered in the graph results in making all inferences regarding time bound values during graph construction. The result is that at any time, for any node t_j in the graph, t_j.lower is greater than or equal to all other lower bounds t_i.lower, where t_i is an ancestor of t_j. Also, t_j.upper is less than or equal to all other upper bounds tk.upper, where t_k is a descendant of t_j.

A recursive propagation procedure is proposed to maintain these relations every time a new bound is entered into the graph. The basic structure of the procedure is as follows:

```
:if propagating a lower bound from node $t_j$ then
    for each of the direct descendants $t_k$ of $t_j$
        if $t_k$.lower < $t_j$.lower then
            set $t_k$.lower := $t_j$.lower
            recursively call this routine with node $t_k$
otherwise (propagating an upper bound)
    for each of the direct ancestors $t_i$ of $t_j$
        if $t_i$.upper > $t_j$.upper then
            set $t_i$.upper := $t_j$.upper
            recursively call this routine with node $t_i$
```

This is a simplified description of the routine since there is no allowance for durations that may exist on the edges between the bounds. Now the reasoning behind

maintaining an ancestor list at every node becomes clear. If the ancestor list is excluded, propogating an upper bound up through the graph becomes time consuming. It would be necessary to check every node in the graph to find all the ancestors of any particular node. This would mean a worst case performance of $O(n^2)$ which is unacceptable.

Suppose that two consecutive nodes t_1, t_2 in the graph have known lower and upper bounds l_1, u_1 and l_2 and u_2 respectively, and further that the time span $t_2 - t_1$ has known lower and upper bounds l and u respectively. Then the inequalities directly bounding t_1 and t_2 and those relating the bounds are as follows:

1) $l_1 \leq t_1 \leq u_1$

2) $l_2 \leq t_2 \leq u_2$

3) $l \leq t_2 - t_1 \leq u$

4) $u_1 \leq u_2$

5) $l_1 \leq l_2$

6) $t_1 \leq t_2$

Now suppose that one or more of the bounds l_1, u_1, l_2, u_2, l, u is updated. Then the following alternative bounding inequalities can be used to update all the pairs of bounds optimally:

7) $l_2 - u \leq t_1 \leq u_2 - l$

8) $l + l_1 \leq t_2 \leq u + u_1$

9) $l_2 - u_1 \leq t_2 - t_1 \leq u_2 - l_1$

For optimal updating, the inequality amoung (1)-(3) and (7)-(9) that provides the best bound on a time is the one

applied in that case. The proof that (7)-(9) are the best supplementary bounds that can be derived from (1)-(6) is provided in Appendix A. The equations are used within the propagation algorithm to always propagate the best possible bound. Also they are applied whenever a new duration is added to the graph, to update the absolute times of the nodes delimiting the duration. If these times can be improved as a result of the new duration, then this is done and the new time is then propagated through the time graph.

Although new durations are used to update time bounds on the nodes of the graph, new time bounds are not used to update the durations. Updating the new time bounds is necessary since a change in time bound can have effects throughout the graph. A new duration could possibly affect the time bounds of a large number of nodes, quite distant in the graph. However, if a new time bound results in a lower or upper bound duration, then this effects nothing but that duration. Recording the new bounds is unnecessary since this would merely duplicate information already directly available to the question answering algorithms.

However, if a new time bound results in a better lower or upper bound duration, then it is sufficient to propagate the effects on the time bounds through the graph. Once this has been done any changes in durations can be calculated locally from these new time bounds.

In order to maintain all given time durations it may be necessary to create new edges. If a duration is specified

between two nodes t_i and t_j, and t_j is a descendant of t_i but there exists no edge directly connecting the two nodes, then an edge is created (t_i, t_j) and the duration is placed on this edge. Notice that if t_j is not a descendant of t_i then the information is not added to the graph. This means that a new duration cannot imply an ordering of time points that did not exist before the introduction of that duration. This forces all ordering relations to be explicitly stated, and hence reduces the possibility of invalid inferences based on implications from durations.

4.3 Time and Space Complexity Analysis

This section examines the time and space complexity for graph construction. The graph and all supporting structures are shown to require O(n + e) storage. The graph construction algorithms are shown to require worst case O(n) time for adding a new event or relation (but constant time in the usual case), and worst case O(n + e) time for adding a new absolute time or duration to the graph. In the following detailed assessment of the complexity, definitions from the start of chapter 4 will be used.

The requirements for the time graph itself represents the major demand on storage. The time graph requires n + e storage just for the nodes and edges. In addition, although it is a directed graph, backward edges are also maintained to allow the propagation algorithm to execute efficiently. This adds another e storage locations. The two labels on

each node adds 2n, and finally the absolute time bounds on each node and edge add a further 2n + 2e storage locations. The total storage required by the time graph alone is (5n + 3e).

There are a number of structures external to the time graph which must also be considered. The first of these is the mapper which holds the mapping between events and time points. This structure will eventually be maintained outside of the time module altogether to allow the general knowledge system free access to it. But it contains information required by the time handler so it will be consistent to include it with these calculations. The worst case for the mapper occurs if there are a maximum n/2 events, with each represented by two time points, using 3n/2 storage. Also, the minimum and maximum label-value for each chain is maintained, which requires 2K storage. Finally, the meta-graph contains a list for every chain (K) plus every cross-chain connection, requiring (K + m) storage.

So the total storage requirements for all the structures is (13n/2 + 3e + 3K + m). Since K < n and m < e, and it is expected that for most graphs K << n and m << e, O(n + e) storage is required.

The time requirements for graph construction can be broken into two components. The first is adding new events or relations. Normally this be performed in constant time, but the worst case is O(n) which occurs if adding a new event requires renumbering a chain. A chain can have at most

n nodes in it, so this requires at most n operations. The most frequent renumbering occurs if each new event added is 'during' the previous one (for example, E_2 during E_1, E_3 during E_2, E_4 during E_3, etc.). Using the current numbering scheme of assigning one tenth of the interval to the new node and assuming the pseudotimes have a maximum of six decimal places, then the tenth such addtion (i.e. E_{10} during E_9) would force a renumbering of the chain.

The second component is adding a new absolute time or duration. In the worst case the propagation of the new time could require traversing the entire graph. The traversal requires e operations, and the comparison of time bounds at every node would add a further O(n) comparisons. The worst case for the addition of a new absolute time or new duration is O(n + e).

Therefore the graph construction has a worst case complexity of O(n + e) for both time and space. However, in narratives that have been represented (such as Little Red Riding Hood) there are few (none for LRRH) absolute times or durations, so the potentially most costly operation (propagation of time bounds) is rarely evoked[11]. This means that updating the graph usually takes place in constant time.

At any rate, this deals only with the issue of constructing the graph. The more important complexity

---

[11] Also, since people are not very good at propagating time bounds this is an "extra" over the type of information that was originally to be the focus of the representation.

measures for the question answering algorithms are given at the end of the next chapter, after a description of those algorithms.

5. Retrieving Temporal Knowledge and Relations

The focus of this chapter is on extracting information from the time graph. The first section describes the inference mechanisms and algorithms for answering questions about relative positioning of time points and events. The second section provides a detailed account of question answering based on dates and durations including inferences and retrieval of information. The third section describes some external controls for limiting the search, and for providing more information in answering a question. An analysis of the time complexity of the algorithms outlined is provided in the fourth section, while the final section provides some empirical results for question answering based on an implementation ofthe time graph outlined here.

5.1 Inferring Relative Position

As mentioned in chapter 3, one of the main goals of this time processor was to be able to perform fast question answering with particular emphasis on the determination of relative order of events. This section describes the algorithms for achieving this based on the time graph structures outlined in chapter 4. Some of these algorithms have been sketched earlier, but will be described more fully here.

The kinds of questions which are of greatest interest here are those which people can generally answer quickly. A story such as "The Old Man and the Sea", by Hemingway [1935]

contains some references to specific times. However, although there are only a few specific times mentioned and thousands of events temporally related by relative position only, the order of events is usually easily distinguished, whereas association of events and times requires more thought. Given two events in the story determination of relative order can usually be done with little or no hesitation. This is apparently independent of the separation of the two events within the story, implying some form of constant time process. While perhaps not difficult, asking for the specific time of an event seems to require more thought than simple ordering. These observations led to the representation described in chapter 3 and 4.

Like the discussion of the graph construction algorithms, this discussion starts with time points, laying the foundation for subsequent discussion of events.

For two time points t_i and t_j, the desire is for constant time determination of relative order of t_i and t_j. However, since this appears to be unattainable, an algorithm guaranteed to complete within linear time and which operates within constant time in the usual case is sought.

The strongest statement inferrable from a time graph about the relative order of two time points t_i, t_j is always one of the following:

1). t_i **Before** t_j

2). t_i **After** t_j

3). t_i **Equal** t_j

4). relation between t_i and t_j **Unknown**

Using notational conventions introduced in chapter 3, the order of t_i and t_j implicit in the structure of the time graph (disregarding time bounds) is determined by the following algorithm:

```
if ($t_i$.chain = $t_j$.chain) then
    if ($t_i$.pseudotime < $t_j$.pseudotime) then
        $t_i$ BEFORE $t_j$
    else
        if ($t_i$.pseudotime > $t_j$.pseudotime) then
            $t_i$ AFTER $t_j$
        else
            $t_i$ EQUAL $t_j$
else ($t_i$ and $t_j$ belong to two different chains)
    search the metagraph from $t_i$.chain and $t_i$.pseudotime
    for $t_j$.chain and $t_j$.pseudotime;
    if found then $t_i$ BEFORE $t_j$
    else
        search the metagraph from $t_j$.chain and
        $t_j$.pseudotime for $t_i$.chain and $t_i$.pseudotime;
        if found then $t_i$ AFTER $t_j$
        else
            $t_i$, $t_j$ UNKNOWN
end;
```

The only part of the algorithm requiring further explanation is searching the metagraph. This search procedure is a recursive one which requires the chain and pseudotime where the search begins, and the chain and pseudotime being searched for. The search algorithm checks all descendant edges of the starting meta-node for the required chain. The following is a detailed description of the search algorithm, given a starting node with chain i and pseudotime m, and searching for chain j and pseudotime n. In the following 'node' refers to a node in the metagraph, and 'AnscVal', 'DescVal' refer to the pseudotimes of the two chains in the time graph where the connection occurs.

```
for each edge $e_i$ connecting a descendant node $N_k$ of the
current node do
    if ($N_k$.chain = j) then
        if ($m \le$ AncsVal) then
            mark $e_i$ as visited
            if (DescVal $\le n$) then
                found := true;
if not found then
    for each edge $e_j$ connecting a descendant node $N_k$ of
    the current node
        if ($N_k$.chain $\neq$ j) and (not visited ($e_i$)) and
        (AncsVal $\ge m$) then
            mark $e_i$ as visited
            found := (Search metagraph from $N_k$.chain and
```

```
            DescVal for chain j and pseudotime n);
    Search := found;( - return value for the function)
    end;
```

To illustrate the process consider a sample time graph with corresponding metagraph, as given in Figures 7 and 8. Now consider the following questions concerning this time graph:

(Relation between 1 and 3?):

- (1.chain = 2.chain) and (1000 < 3000) ⇒ 1 *BEFORE* 3;

(Relation between 1 and 7?):

- search descendants of A (B, D) and find B
- (connection at A(2000) > pseudotime for node 1(1000)) and (connection at B(1000) < pseudotime for node 7(2000)) ⇒ search successful

Hence, 1 *BEFORE* 7;

(Relation between 3 and 7?):

- search descendants of A (B, D) and find B (connection at A(2000) < pseudotime for node 3(3000)) ⇒ search failed
- search descendants of B looking for A
- this search fails as well

Hence, *UNKNOWN*

(Relation between 8 and 13?):

- search descendants of C (A)
- recursively search from A at 3000 for E at 2000
- descendants of A (B, D)

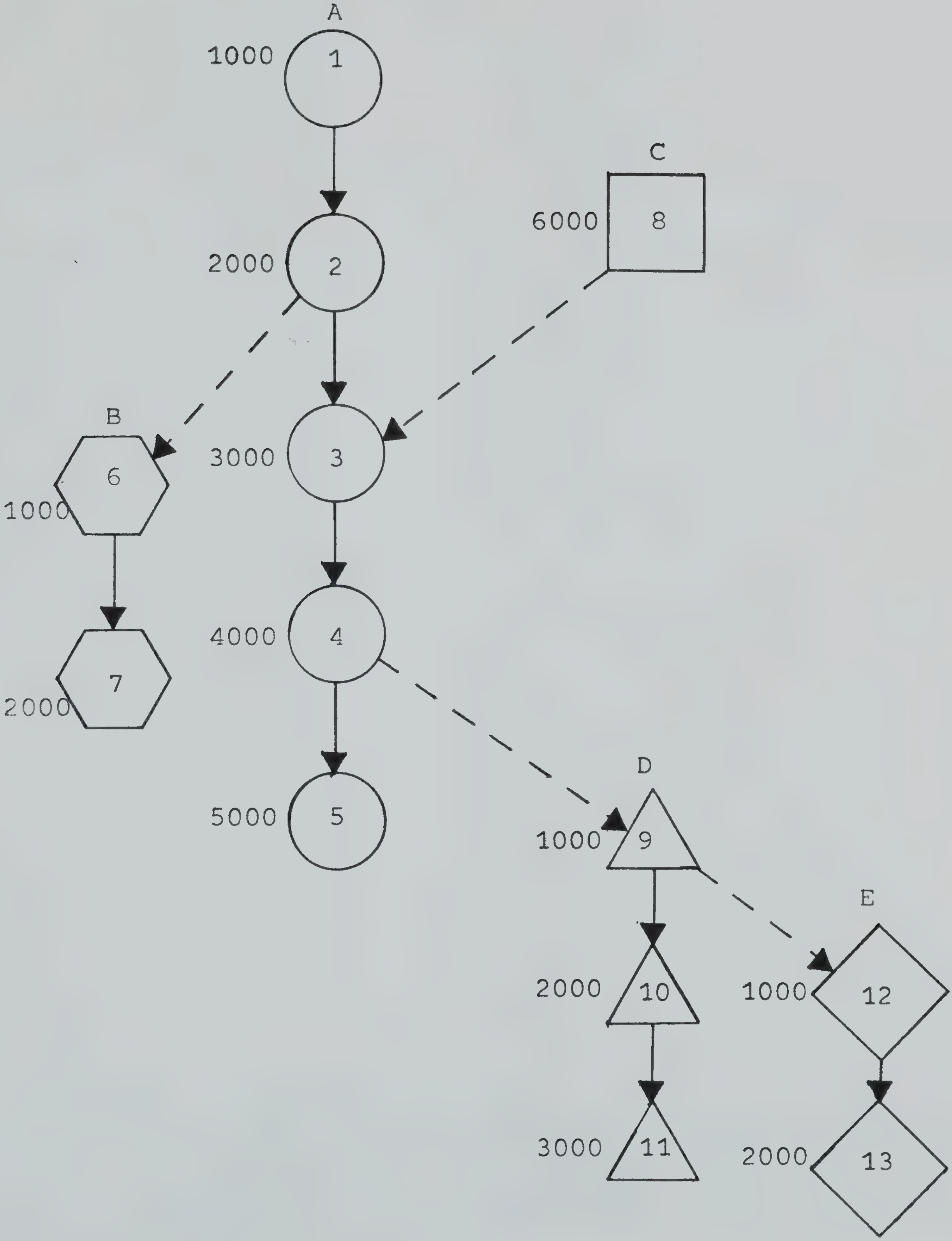

Figure 7: A Fully labelled Time Graph G_2.
The numbers within each node record the narrative sequence, i.e., the order in which the nodes were added. Nodes of the same shape belong to the same chain (marked with a letter). The numbers adjacent to each node are the pseudotimes of each node.

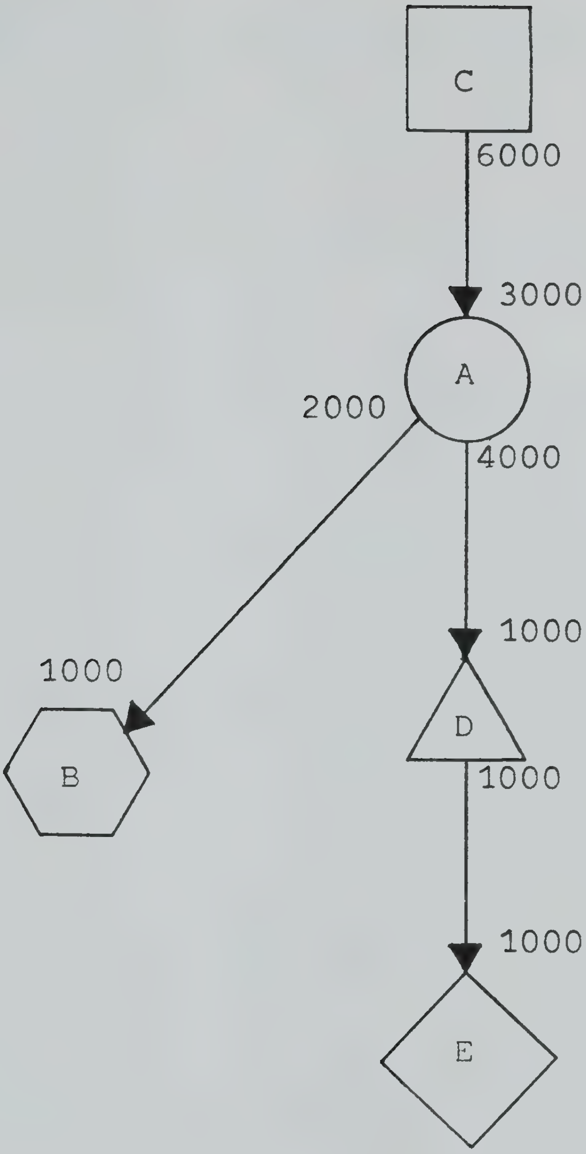

Figure 8: The Metagraph for G_2.
The nodes of the metagraph correspond to the chains in the time graph G_2. The pseudotimes marked beside the head and tail of each edge indicate the pseudotimes of the node within the chain where the connection occurs.

- search chain B at 3000 for E at 2000
  failed
- search chain D at 1000 for E at 2000
  - descendants of D are (E)
    - success
- the success is returned through all the calls and hence, 8 *BEFORE* 13;

Since the metagraph is not acyclic and since a single chain might be a descendant of many other chains, the search of the metagraph marks each edge as visited once it has been examined. This prevents recursive calls from searching through edges already examined and discarded.

This completes the discussion on determination of relative position of nodes within the graph, from which the relative order of time points can be inferred. Relations between events can be determined by breaking an event query into a sequence of queries regarding the constituent time points of the events. Consider a query of the relative order of events E_i and E_m with constituent time points t_i, t_j and t_m, t_n respectively. The following are the relations between the two events that appear to be the most important in practice:

1) E_i **Before** E_m
2) E_i **After** E_m
3) E_i **Equal To** E_m
4) E_i **Contains** E_m
5) E_i **During** E_m

6) E_i **Overlaps** E_m

7) E_i **Overlapped By** E_m

Each of these can be broken into one or more time point relations, which imply the event relation. These are:

1) t_j Before t_m

2) t_i After t_n

3) (t_i Equal t_m) and (t_j Equal t_n)

4) (t_i Before t_m) and (t_n Before t_j)

5) (t_i After t_m) and (t_n After t_j)

6) (t_i Before t_m) and (t_m Before t_j) and (t_j Before t_n)

7) (t_i After t_m) and (t_n After t_i) and (t_j After t_n)

By forming the queries in this way all the processing is done by the time point algorithms. However, in order to claim any of the seven event relations, all of the above conditions for that relation must hold. But where there is more than one constituent time point relation, the requirements may be only partially fulfilled. In such cases, there may be no event relation which is fully defined, but there may still be some valuable information provided by response to the time point queries. In order to transmit as much information as possible a further two relations are defined to cover any partially defined event relations. These are:

8) E_i **Starts Before** E_m (starts)

9) E_i **Ends Before** E_m (ends)

10) E_i **Starts same time as** E_m (starts)

11) E_i **Ends same time as** E_m (ends)

12) E_i and E_m **are consecutive**

These extra relations (and their inverses) may be used when none of the fuller definitions are satisfied but there are one or more pairs of time points whose relation is known. In this way, whatever information is available concerning the relative position of the two events can be provided.

The algorithm for answering questions about events simply forms four time point queries from the two events in question. The responses to the time point queries are then used to try to satisfy one of the seven major event relations. If this fails then the algorithm checks the auxilary relations to see if one of them can be satisfied. If no event relation can be satisfied then a response of "unknown" is returned.

5.2 Inferences Based on Absolute Times or Durations

In this section, the use of absolute times and durations to determine relative positions of time points and events is outlined. Also, the methods for extracting dates and durations themselves are detailed.

As described in 4.2.2 absolute times are propagated through the time graph providing improved bounds wherever they can be inferred. Also, durations are used to improve the absolute times so that at any node in the graph the time bounds on that node are the best that can be inferred from information presented. This is done to provide quick access

to absolute time bounds which speeds question answering concerning these times.

In addition to the methods described in the previous section, absolute times can be used to determine the relative position of two nodes in the graph. Consider time points t_1 and t_2 with lower and upper bounds l_1, u_1 and l_2, u_2 respectively. If $l_1 > u_2$ then t_2 is before t_1, and similarly if $l_2 > u_1$ then t_2 is after t_1. Thus two time bound comparisons may yield the relation between two time points. For this reason, comparison of the absolute times on two nodes is performed prior to a search of the metagraph. However, this check is performed after the check for t_1, t_2 on the same chain, since for some graphs there is little or no absolute time information.

This quick check on relative position is a fringe benefit of maintaining absolute times. The main reason they are kept updated is to allow fast retrieval of time bounds for any time point or event. As with determining relative position, determining absolute time bounds for events is done by first determining the bounds on the constituent time points, and then combining this information to infer the bounds on the event.

Queries concerning the dates of time points are satisfied by retrieving the absolute time bounds present on the corresponding node in the time graph. If there is no information on one of the bounds, then a value of "unknown" is returned for this bound. Otherwise, whatever information

is present[12] can be provided in response to the query.

After retrieving the time bounds of the time points composing an event, an upper bound on the absolute time of an event is obtained by using the lower bound of the first time point and the upper bound of the second time point. If more information is required, the best that can be done is to retrieve all the absolute time information from the two time points composing the event. So if the time bounds on t_1, t_2 are l_1, u_1 and l_2, u_2, then "event starts between l_1 and u_1 and ends between l_2 and u_2" is the most information that can be retrieved.

There are two methods employed for retrieving time durations for events or between pairs of time points or events. The first and simplest method is to check the edge, if any, between the two nodes in the graph which delimit the requested duration. If the duration concerns an event E_i with end points t_1 and t_2, then the required edge will connect t_1 and t_2. If the duration is between two time points t_i, t_j then the edge must connect t_i to t_j, and if the duration is between two events E_i, E_m with time points t_i, t_j and t_m, t_n then the edge must connect t_j to t_m. If such an edge is found, then it is examined to determine if there is duration information recorded on this edge. If the information is present, then this can be immediately extracted to provide an answer to the question. If not present, then a second algorithm is initiated to attempt to

---

[12] This may include constants or partially specified dates such as just the year, month and day only, etc.

derive the duration from absolute time information.

Given two time points t_1 and t_2 as described above, a duration between t_1 and t_2 can be inferred from the absolute time information on these nodes. If lower and upper bounds are present on both time points then the duration is derived using the equations given in 4.2.2. The lower bound on duration is $l_2 - u_1$ (if $u_1 \geq l_2$ then 0), while the upper bound is $u_2 - l_1$ (l_1 cannot be larger than u_2, since $l_1 \leq u_1 \leq u_2$). These are the best bounds that can be inferred from the absolute times on the nodes[13] assuming that all the time bounds are present, and that they are comparable.

If for any reason these bounds cannot be calculated, then there exists a weaker value for each bound which may be used, although for practical purposes it is unlikely to be useful. Of course, in the absence of this information, a value of "unknown" must be returned.

To illustrate how duration information is extracted, consider Figure 9. A query concerning the duration between time points t_1 and t_2 can be answered from two sources. First, there may be durational information (l_4, u_4) on the edge connecting the two time points. Second there may be durational information implicit in the absolute time bounds of the two time points. When such a competition exists, the best bound is chosen. In this case, l_4 = 2 months, and $l_2 - u_1 < 0$, so 2 months is the best lower bound. Similarly, u_4 = 4 months, and $u_2 - l_1$ = 9 months and 2 days, so 4 months is

---

[13] For proof of this see Apendix A.

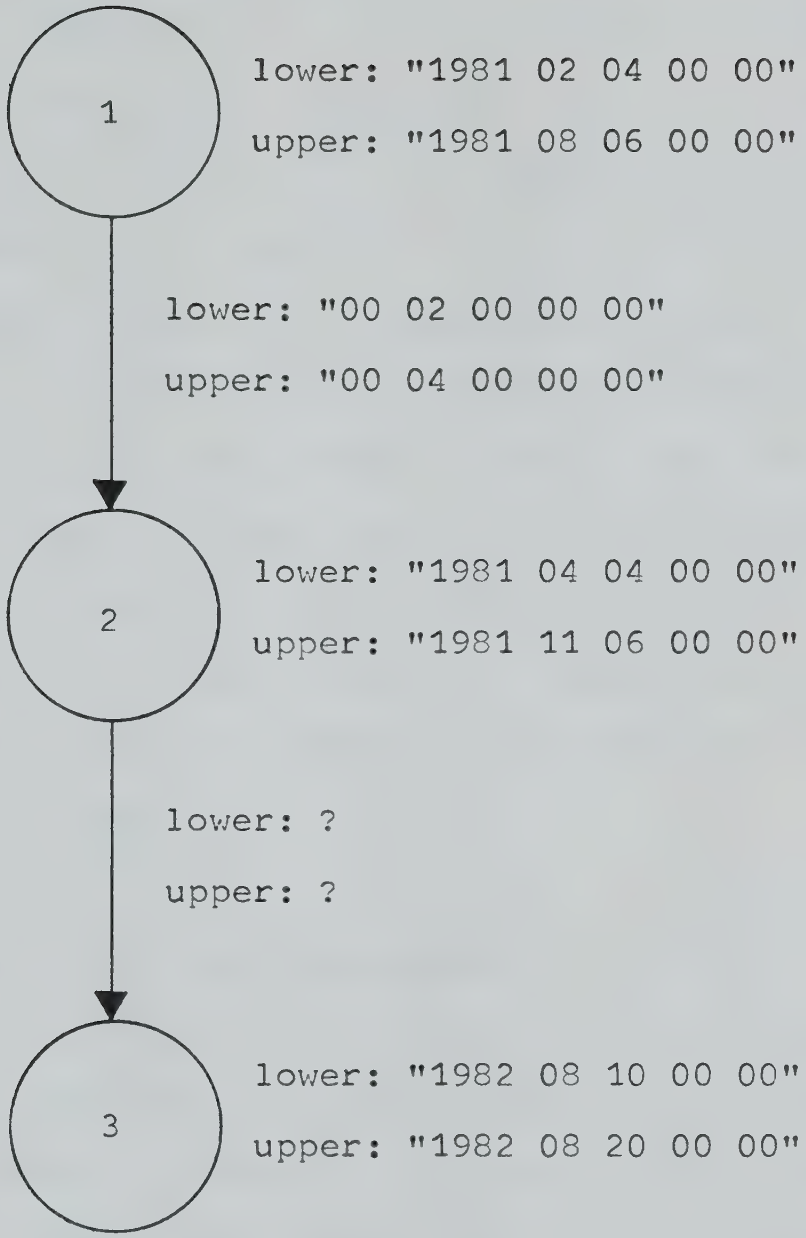

Figure 9: Absolute times and Durations on a time graph G_3.
The lower and upper bounds for the absolute times are printed beside the appropriate node, with the duration values marked on the edges.

the better upper bound. Hence the response would be, "between two and four months".

To illustrate how this might change, consider subsequent changes to the absolute time bounds such that l_2 = "1981 10 16 00 0.0" and l_1 = "1981 08 01 00 0.0". Now $l_2 - u_1$ (2 months and 10 days) > l_4 and so provides a better lower bound, and $u_2 - l_1$ (3 months and 5 days) < u_4 and hence is a better upper bound. A query concerning the duration between time points t_1 and t_3 must rely on the absolute times on these nodes. In this case (returning to the original Fig. 7), the lower bound would be $(l_3 - u_1)$ and the upper bound would be $(u_3 - l_1)$ making the response, "between one year and four days, and one year, six months, and sixteen days".

5.3 External Controls on Question Answering

This section describes controls over the question answering algorithm. The first mechanism controls the maximum amount of effort to be expended on answering a particular query. The second controls the amount of information provided in response to a query. The final mechanism controls error checking on new information.

Special purpose inference methods such as the time representation and inference algorithms outlined in this thesis are designed to augment the general reasoning process. The general reasoning system consults (poses questions to) the special purpose inference mechanisms in

order to achieve fast resolution of problems in that domain. However, since the special purpose methods are to be subservient to the general reasoning system, there needs to be some provision for control over the length of time spent on answering a particular question.

A parameter can be included with the only type of question that can incur worse than O(1) time, questions concerning relative position. The parameter indicates the level of searching that should be attempted before aborting the question answering process. The top level specifiable places no restrictions on the question answering algorithms. This is used when resolving a question takes priority over time spent by the algorithms. Lower levels indicate successively tighter restrictions on time consumption. The lowest level limits the algorithms to constant time processes[14] which assure a response (whether or not the question has been successfully answered) in constant time. Of course, whenever an answer is successfully derived from the time graph, the answer is returned and the controls on search levels have no effect.

While providing correct answers to questions is the primary function of these methods, it is easy to envisage a situation where providing an answer is not sufficient. For example, a situation in which the answer to a question is challenged by the further question, "How do you know that?". If possible within the framework already outlined, a method

---

[14] The metagraph is never consulted at this level.

of indicating how an answer was obtained would be beneficial. A second control parameter is provided which creates a trace[15] of the derivation of an answer.

This parameter can be set when a question is presented to the time handler, and if an answer is found, both the answer and the sequence of inferrences that were made in forming the answer are returned. This is done by recording the search through the metagraph. Everytime a new node in the metagraph is examined an entry is added to the trace. If the search fails at a particular node and the algorithm backtracks to try a new node, then the failed node is removed from the trace. If the search is successful then the trace contains the sequence of nodes in the metagraph that form a path between the two nodes inquired about. This constitutes the inter-chain connections which are followed to find a path in the time graph. Sequences of inferences within a single chain can be related easily by simply following the nodes in the chain, so no special method is needed for these inferences.

A final parameter is included to control the level of error checking carried out by the graph construction algorithms. This consists of two levels indicating whether or not new information is considered reliable. If unreliable, new information that causes internal contradictions is not added to the graph.

---

[15] For questions concerning relative position, which can involve the only long sequences of inferences.

In this way some degree of control is provided to the system using this special purpose representation. The control covers the amount of time spent on trying to find an answer, and the amount of information provided by the answer. Both of these features are essential to mold this system into the framework of a general reasoning process.

5.4 Time and Space Complexity Analysis

This section examines the theoretical time requirements of the question answering algorithms. This analysis uses the descriptions and definitions given in chapter 4. The analysis of the storage requirements for all the structures used in this representation, along with the time complexity analysis of the graph construction algorithms have been presented in 4.3. At the outset, the analysis ignores the effect of the above mentioned controls on the question answering algorithms.

The questions whose time requirements are of greatest concern are those about the temporal relation between two time points. The worst case for this type of question could involve searching the entire metagraph. Consider such a query concerning two time points t_i and t_j.

The worst case can occur if t_i and t_j are incomparable. In this case the algorithm tries first to find t_j as a descendant of t_i, and on failure tries to find t_i as a descendant of t_j. It is only when both of these attempts fail, that the algorithm can respond with the correct

relation, "Unknown".

The algorithm which searches the metagraph, marks nodes as they are searched so that no node is searched twice. Since the metagraph consists of K nodes and m edges, searching the entire graph requires O(K + m) comparisons. The worst case requires searching the metagraph twice and hence still O(k + m) comparisons. The original check for identical chains requires a single comparison, and resetting the metagraph edges in preparation for the next search (which must be done twice) has a time requirement of O(K). So the worst case for determining relative position of two nodes within the graph is O(K + m).

This satisfies questions regarding the relative order of pairs of time points, but not events. However, since event queries are simply broken into time point queries, no new analysis is necessary. If comparison of events E_1 and E_2 composed of time points t_1, t_2 and t_3, t_4 is requested, the algorithm formulates at most four time point queries (t_1,t_3; t_1,t_4; t_2,t_3; t_2,t_4) in order to derive the relation between E_1 and E_2. So this increases the time requirements by a factor of four, leaving the order of complexity unchanged.

The complexity remains unchanged when the controls of the previous section are introduced. A parameter limiting the amount of searching can only speed the time requirements of question answering. The trace of the path found between two nodes requires recording each node in the metagraph as it is searched, and erasing each node which fails. This

could involve worst case 2K steps, which leaves the time complexity at O(K + m).

Questions involving absolute times or durations can all be answered in constant time, since no searching is required. Therefore, the worst case time requirements for the entire question answering algorithms is O(K + m). Note that this measure is independent of *n*, the number facts stored.

5.5 Empirical Results

In this final section some empirical results from question answering on the time graph are presented. The time graph outlined in this thesis was implemented in Pascal on a Digital Vax 11/780. Since the relative order of time points is the primary focus for efficiency, time graphs excluding absolute times and durations were constructed for these tests. One of these test graphs (G_3 in table 1) represents the story Little Red Riding Hood, while the rest were artificially constructed to manipulate n, m and K.

The results are broken into two groups, with all question answering times (in seconds) based on 1500 randomly chosen relative time point order queries. In the first group (table 1), the time graphs vary in size from 30 to 1200 nodes and the question answering algorithm is shown to be independent of the number of nodes in the graph. Three of the graphs G_4, G_5, G_6 have n ranging from 300 to 1200 but because the graph partitioning (K + m) is roughly constant

Time Graph	n	K	M	K+M	Total Time (sec.)	Read Graph (sec.)	Net Q.A. Time
G_1	30	4	3	7	25.2	4.6	20.6
G_2	100	18	27	45	55.2	5.8	49.6
G_3	290	21	33	54	53.3	8.3	45.0
G_4	300	15	15	30	43.0	9.2	33.8
G_5	600	15	14	29	46.1	15.1	31.0
G_6	1200	15	14	29	60.6	28.3	32.3

Table 1
Question answering times on a Vax 11/780 for 1500 time order questions, for graphs containing 30 - 1200 time points.

Time Graph	n	K	M	K + M	Net Q.A. Times (sec.)
G_1	600	15	14	29	27.7
G_2	600	23	22	45	36.0
G_3	600	30	29	59	45.2
G_4	600	46	45	91	55.1
G_5	600	45	61	107	63.1

Table 2
Question answering times on a Vax 11/780 for 1500 time order questions, for graphs with a fixed number of time points (600) and variable K + M.

over these three graphs, the question answering times also remain constant over these three graphs.

In the second group of tests (table 2), the number of nodes in the time graph was kept constant ($n = 600$) and the values of $K + m$ are varied to illustrate the question answerers' dependence on this factor (and not n). The addition of absolute times and durations to these graphs cannot worsen the question answering times since they can simply be ignored in answering relative order questions, and in fact will improve the speed since occasionally absolute time bound specifications will allow determination of relative order and thereby avoid the metagraph search.

6. Conclusions

6.1 Results

There has been a growing recognition in past years that reasoning about everyday objects and events is not practical using uniform inference methods. Unless special purpose inference methods are developed for certain important classes of relations, such as relations over concept types, over parts of objects, and over times, the reasoning needed to support natural language understanding and question answering will be unacceptably slow. The proposed representation demonstrates that specialized representations can be designed to augment general reasoning processes, and allow inference mechanisms to efficiently handle problems that would otherwise require large computational resources.

The scope of the time specialist could be broadened to handle other types of questions. Kahn's system [1975] had the ability to answer questions such as, "What happened during April 1963?", which are beyond the present capabilities of the system presented here. However, this type of question is rather hard for people and the present system focuses on areas in which people are proficient. In those areas, it achieves unprecedented efficiency.

A lesson learned from this investigation is that (in the representation field at least) analogies can be deceptive. In particular, the fact that containment, overlap and disjointness relations amoung intervals are analogous to

similar relations amoung concept types and amoung parts of objects led to a misguided attempt to apply P-graphs to time inference. The analogy fails because the additional properties of time intervals in particular the underlying linear structure of time permit the use of a more efficient representation exploiting these properties. An attempt to apply P-graphs to colour inference had proved unsatisfactory for similar reasons.

While the temporal representation described here was designed primarily to match the cognitive proficiencies of people it was augmented with methods for inferring absolute time bounds and durations even though people are weak in this area. This was done simply because relatively little extra work was required to add this (potentially quite useful) capability to the system. Thus the question answering abilities of this temporal processing system exceed the original goals.

6.2 Future Research

Since this time module is intended to deal with time inferences in a general language translation system, there will be widely divergent or even contradictory facts from different stories or between fiction and non-fiction events. In order to deal with such situations, it will be necessary to create distinct time graphs to correspond to distinct modal contexts (especially belief and story contexts) in the general knowledge system. This would require some

modifications to the representation proposed here. A method of dynamically creating a new time graph with all of the supporting structures would be required. Also, some new methods for resolving inter-graph queries would be necessary.

An area requiring more effort in both the temporal representation and the knowledge system as a whole, is error checking and correcting. Although it is possible to check for contradictions with existing information whenever new information is added, there is no method of determining where to place the blame if a contradiction is found. In the proposed representation in particular, inferences are sometimes made when a new fact is introduced which would have to be retracted if the fact was discovered erroneous.

What is required to solve this problem is a means of specifying a range of confidences and attaching these to each fact in the system. This would aid the resolution of contradictions by giving the reasoning system some means of judging which of a set of conflicting facts and infereneces are to be believed. Whenever conflicts occured, the algorithms would choose the information with the highest confidence level.

Another possibility is to have all such conflicts which occur in the time handler referred to the general reasoner for resolution. There may be other facts in the system which would enable a decision on the relative validity of the

information[16].

A final modification that deserves consideration is the addition of capabilities for answering questions partially. Many times the question answerer will come close to finding an answer, but must return "unknown". There should be some facility for indicating to the question asker how close to providing an answer the algorithm was. Even more beneficial, although more difficult, would be an indication of what information is needed, or where the algorithm failed. If this was provided to the general knowledge system, it could be used to derive more information concerning the events from world knowledge.

---

[16] Since the general reasoner would have access to world knowledge, other special purpose domains, etc.

Bibliography

Allen, J. F. (January 1981). "Maintaining Knowledge about Temporal Intervals." *Univ. Rochester Tech. Report,* (TR-86).

Allen, J. F. (November 1981). "A General Model of Action and Time." *Univ. Rochester Tech. Report,* (TR-97).

Allen, J. F. and Koomen, J. A. (1983). "Planning Using a Temporal World Model." *Proc., 8th IJCAI, Karlsruhe, W. Germany., Aug. 8-12, 1983.* pp. 741-747.

Almeida M. J. and Shapiro, S. C. (1983). "Reasoning About the Temporal Structure of Narrative Texts." *5th Ann. Conf. of the Cognitive Science Soc.* Session 5, pp. 1-5.

Baker, K. A., Fishburn, P. C. and Roberts, F. S. (1979). "Partial Orders of Dimension 2." *Networks 2,* pp. 11-28.

Bruce, B. C. (1972). "A Model for Temporal References and Its Application in a Question Answering Program." *Artificial Intelligence. Volume 3.* pp. 1-25.

Bruce, B. C. (1973). "The Processing of Time Phrases in Chronos." *Rutgers Univ. Tech. Report,* (CBM-TR-29).

Bruce, B. C. and Singer, N. (1972). "Chronos-2: The processing of Incompletely specified Data." *Rutgers Univ. Tech. Report*, (CBM-TR-9).

Cohen, R. (1977). "Computer Analysis of Temporal Knowledge." *Univ. Toronto Tech. Report*, (TR-107).

Covington, A. R. and Schubert, L. K. (1979). "Organization of modally embedded propositions and of dependent concepts." *Proc. 3rd Bienn. Conf. of the CSCSI/SCEIO, Victoria, B.C., May 14-16, 1980.* pp. 87-94.

Dushnik, B. and Miller, E. W. (1941). "Partially Ordered Sets." *Amer. J. Math. 63, 1941.* pp. 600-610.

Findler, N. V. and Chen, D. (1971). "On the Problems of Time, Retrieval of Temporal Relations, Causality, and co-existence." *Proc. 2nd IJCAI 1971.* pp. 531-545.

Grover, M. D. (1982). "A synthetic Approach to Temporal Information Processing." *Proc. AAAI 1982, Vol. 1* pp. 91-94.

Hemingway, E. (1952). *The Old Man and the Sea.* Published by S. J. R. Saunders and Co. Ltd.

Hirschman, L. (1981). "Retrieving time information from natural-language texts." *Information Retrieval Research*, Buttersworth, London (1981). pp. 154-171.

Hirschman, L. and Story, G. (1981). "Representing Implicit and Explicit Time Relations in Narrative." *Proc. 7th IJCAI, Vancouver, B.C., Aug. 24-28, 1981*. pp. 289-295.

Kahn, K. (1975). "Mechanization of Temporal Knowledge." *MIT Tech. Report*, (MAC-TR-155).

Kahn, K. and Gorry, G. A. (1977). "Mechanizing Temporal Knowledge." *Artificial Intelligence 9, (1977)*. pp. 87-108.

Kameda, K. (1975). "On the Vector Representation of the Reachability in Planar Directed Graphs." *Information Proc. Letters, Vol. 3, Jan. 1975*. pp. 75-77.

Kandrashina, E. Y. (1983). "Representation of Temporal Knowledge." *Proc. 8th IJCAI, Karlsruhe, W. Germany, Aug. 8-12, 1983*. pp. 346-348.

Malik, J. and Binford, T. O. (1983). "Reasoning in Time and Space." *Proc. 8th IJCAI, Karlsruhe, W. Germany, Aug. 8-12, 1983*. pp. 343-345.

McDermott, D. (1982). "A Temporal Logic for Reasoning about Processes and Plans." *Cognitive Science 6, (1981)* pp. 101-155.

Papalaskaris, M. A. (1982). "Special Purpose Inference Methods." *Univ. of Alberta, M.Sc. Thesis (1982).*

Papalaskaris, M. A. and Schubert, L. K. (1981) "Parts Inference: Closed and Semi-Closed Partitioning Graphs." *Proc. 7th IJCAI, Vancouver, B.C., Aug. 24-28* pp. 304-309.

Papalaskaris, M. A. and Schubert, L. K. (1982) "Inference, Incompatable Predicates and Colours." *Proc. 4th Bienn. Conf. of the CSCSI/SCEIO, Saskatoon, Man. (1982)* pp. 97-102.

Schubert, L. K. (1979) "Problems with Parts." *Proc. 6th IJCAI, Tokyo. Aug.20-23.* pp. 778-784.

Schubert, L. K. (1980) "Representing and Using Knowledge about Parts." *Unpublished Manuscript, Univ. of Alberta,* Edmonton, Alta.

Schubert, L. K., Papalaskaris, M. A. and Taugher, J. E. (1983) "Special Inference Methods in a Semantic Net: Lattices of Types Parts, Colours and Times." *Univ. of*

Alberta Tech. Report, (TR83-3).

Vilain, M. B. (1982). "A System for Reasoning about Time." *Proc. Nat. Conf. AI., AAAI-82, Vol. 1* pp. 197-201.

Appendix A: Proof on Derived Absolute Time Bounds

Five inequalities that maintain consistency relating the lower and upper bounds l_1, u_1, l_2, u_2, l and u of two time points t_1 and t_2 are given. These are:

(1) $l_1 \leq u_1$

(2) $l_2 \leq u_2$

(3) $l \leq u$

(4) $l \leq u_2 - l_1$

(5) $l_2 - u_1 \leq u$

and $(l, u) > 0$

The following new bounds for t_1, t_2, and $t_2 - t_1$ were derived:

(6) $l_2 - u \leq t_1 \leq u_2 - l$

(7) $l + l_1 \leq t_2 \leq u + u_1$

(8) $l_2 - u_1 \leq t_2 - t_1 \leq u_2 - l_1$

The bounds of each variable (t_1, t_2, $t_2 - t_1$) are updated to the best bound between the original and the derived bound (i.e. $\max[l_1, l_2 - u]$ for a lower bound on t_1, $\min[u_1, u_2 - l]$ for the upper bound on t_1, etc.) producing new bounds L_1, L_2, L, U_1, U_2, U.

The method of the proof is to show that a single pass through the inequalities, updating the bounds in turn (in arbitrary order) is sufficient to set every bound to the best possible from the derived inequalities. Secondly, it will be shown that once the new bounds have been set they are the best possible bounds.

After one iteration of the updating cycle the following state exists:

(9)$L_1 = \max[l_1, l_2 - u]$

(10)$L_2 = \max[l_2, L + l_1]$

(11)$L = \max[l, l_2 - u_1]$

(12)$U_1 = \min[u_1, u_2, - l]$

(13)$U_2 = \min[u_2, u + u_1]$

(14)$U = \min[u, u_2, - l_1]$

In order to show that further iterations will not change these bounds, it will be shown that a second update would not alter the bounds above. So for lower bounds it will be shown that a second update would be less than or equal to the first, and for upper bounds it will be shown that a second update would be greater than or equal to the first.

a)Prove that $\max[l_1, l_2 - u] \geq \max[L_1, L_2 - U]$

Expanding the right hand side to get:

$\max[\max[l_1, l_2 - u], \max[l_2, l + l_1] \min[u, u_2 - l_1]]$

$= \max[l_1, l_2 - u, l_2 - u, l_2 - u_2 + l_1, l + l_1 - u, l + l_1 - u_2 + l_1]$

But, $l_2 - u_2 + l_1 \leq l_1$ (since $l_2 \leq u_2$ (2))

Therefore, $l_2 - u_2 + l_1$ can be eliminated from the set.

and $l + l_1 - u \leq l_1$ (since $l \leq u$ (3))

Therefore, $l = l_1 - u$ can be eliminated.

and $l + l_1 - u_2 + l_1 \leq l_1$ (since $l \leq u_2 - l_1$ (4))

Therefore, $l + l_1 - u_2 + l_1$ can be eliminated, leaving $\max[l_1, l_2 - u]$.

Therefore, $\max[L_1, L_2 - U] \le \max[l_1, l_2 - u]$

Hence, one pass is sufficient for L_1.

Using similar techniques on the other bounds,

b) Prove $\max[l_2, l + l_1] \ge \max[L_2, L + L_1]$

R.H.S. $= \max[l_2, l = l_1, \max[l, l_2 - u_1] + \max[l_1, l_2 - u]]$

$= \max[l_2, l + l_1, l + l_1, l + l_2 - u, l_2 - u_1 + l_1, l_2 - u_1 + l_2 - u]$

but, $l + l_2 - u \le l_2$, (since $l \le u$ (3))

and, $l_2 - u_1 + l_1 \le l_2$, (since $l_1 \le u_1$ (1))

and, $l_2 - u_1 + l_2 - u \le l_2$, (since $l_2 - u_1 \le u$ (5))

Therefore, $= \max[l_2, l + l_1]$

c) Prove $\max[l, l_2 - u_1] \ge \max[L, L_2 - U_1]$

R.H.S. $= \max[l, l_2 - u_1, \max[l_2, l + l_1] \min[u_1, u_2 - l]]$

$= \max[l, l_2 - u_1, l_2 - u_1, l_2 - u_2 + l, l + l_1 - u_1, l + l_1 - u_2 + l]$

but, $l_2 - u_2 + l \le l$, (since $l_2 \le u_2$ (2))

and, $l + l_1 - u_1 \le l$, (since $l_1 \le u_1$ (1))

and, $l + l_1 - u_2 + l \le l$, (since $l \le u_2 - l_1$ (4))

d) Prove $\min[u_1, u_2 - l] \le \min[U_1, U_2 - L]$

R.H.S. $= \min[u_1, u_2 - l, u_2 - l, u_2 - l_2 + u_1, u + u_1 - l, u + u_1 - l_2 + u_1]$

but, $u_2 - l_2 + u_1 \ge u_1$, (since $u_2 \ge l_2$ (2))

and, $u + u_1 - l \ge u_1$, (since $u \ge l$ (2))

and, $u + u_1 - l_2 + u_1 \geq u_1$, (since $u \geq l_2 - u_1$ (5))

Therefore, $= \min[u_1, u_2 - l]$

e) Prove $\min[u_2, u + u_1] \leq \min[U_1, U_2 - L_1]$

R.H.S. $= \min[u_2, u + u_1, u + u_1, u + u_2 - l, u_2 - l_1 + u_1, u_2 - l_1 + u_2 - l]$

but, $u + u_2 - l \geq u_2$, (since $u \geq l$ (3))

and, $u_2 - l_1 + u_1 \geq u_2$, (since $u_1 \geq l_1$ (1))

and, $u_2 - l_1 + u_2 - l \geq u_2$, (since $u_2 - l_1 \geq l$ (4))

Therefore, $= \min[u_2, u + u_1]$

f) Prove $\min[u, u_2 - l_1] \leq \min[U, U_2 - L_1]$

R.H.S. $= \min[u, u_2 - l_1, u_2 - l_1, u_2 - l_2 + u, u + u_1 - l_1, u + u_1 - l_2 + u]$

but, $u_2 - l_2 + u \geq u$, (since $u_2 \geq l_2$ (2))

and, $u + u_1 - l_1 \geq u$, (since $u_1 \geq l_1$ (1))

and, $u + u_1 - l_2 + u \geq u$, (since $u \geq l_2 - u_1$ (5))

Therefore, $= \min[u, u_2 - l_1]$

Now from these proofs, the following conditions necessarily apply after updating:

(15) $L_1 = \max[l_1, l_2 - u] \geq \max[L_1, L_2 - U]$

(16) $L_2 = \max[l_2, l + l_1] \geq \max[L, L + L_1]$

(17) $L = \max[l, l_2 - u_1] \geq \max[L, L_2 - U_1]$

(18) $U_1 = \min[u_1, u_2 - l] \leq \min[U_1, U_2 - L]$

(19) $U_2 = \min[u_2, u + u_1] \leq \min[U_2, U + U_1]$

(20) $U = \min[u, u_2 - l_1] \leq \min[U, U_2 - L_1]$

and,

(21) $L_1 \leq t_1 \leq U_1$

(22) $L_2 \leq t_2 \leq U_2$

(23)$L \leq t_2 - t_1 \leq U$

(24)$t_1 \leq t_2$

Now it remains to prove that after updating, each of the bounds (given by (9) - (14)) is the best possible bound. If the variable for each bound is taken in turn, and shown that there exists a choice for the variables with one variable equal to the bound, which is consistent with the updated inequalities ((21) - (24)), then there cannot exist a better bound for that variable.

i) Prove that $t_1 = L_1$ is possible.

Set $t_1 = L_1$, $t_2 = L_2$

Now, (from 21) $L_1 \leq L_1 \leq U_1$ (given)

(from 22) $L_2 \leq L_2 \leq U_2$ (given)

(from 23) $L \leq L_2 - L_1 \leq U$ (from (16), (15))

(from 24) $L_1 \leq L_2$ (from (16) $L_2 \geq L + L_1$, and since $L \geq 1 > 0$)

Therefore, $t_1 = L_1$ is possible (for at least one choice of t_2) and hence provides the best possible bound.

ii) Prove that $t_2 = L_2$ is possible.

Set $t_2 = L_2$, $t_1 = L_1$

This assignment of the variables is identical to the one above and therefore already proved.

iii) Prove that $t_2 - t_1 = L$ is possible.

Set $t_1 = U_1$, $t_2 = L + U_1$

(from 21) $L_1 \leq U_1 \leq U_1$ (given)

(from 22) $L_2 \leq L + U_1 \leq U_2$ (from (17), upper bound

from 18)

(from 23) $L \le L \le U$ (given)

(from 24) $U_1 \le L + U_1$ (since $L \ge 1 > 0$)

iv) Prove that $t_1 = U_1$ is possible.

Set $t_1 = U_1$, $t_2 = U_2$

(from 21) $L_1 \le U_1 \le U_1$ (given)

(from 22) $L_2 \le U_2 \le U_2$ (given)

(from 23) $L \le U_2 - U_1 \le U$ (lower bd. (18), upper bd. (19))

(from 24) $U_1 \le U_2$ (from (18),$U_1 \le U_2 - L \le U_2$ since $L > 1 > 0$))

v) Prove that $t_2 = U_2$ is possible. This is done in (iv).

vi) Prove that $t_2 - t_1 = U$ is possible.

Set $t_1 = L_1$, $t_2 = U + L_1$.

(from 21) $L_1 \le L_1 \le U_1$ (given)

(from 22) $L_2 \le U + L_1 \le U_2$ (lower bd. (15), upper bd. (20))

(from 23)$L \le U \le U$ (given)

(from 24) $L_1 \le U + L_1$ (since $U \ge u > 0$)

Since it has been shown that there exists a valid assignment for each bound, these must be the best bounds that can be assigned to these variables, given the original constraints.

B30389